공학박사 학위논문

A Stacked Memory Architecture for Improving Performance and Capacity

성능과 용량 향상을 위한 적층형 메모리 구조

2019 년 2 월

서울대학교 융합과학기술대학원

융합과학부 지능형융합시스템전공

이 석 한

A Stacked Memory Architecture for Improving Performance and Capacity

지도 교수 안 정 호

이 논문을 공학박사 학위논문으로 제출함
2019 년 1 월

서울대학교 융합과학기술대학원
융합과학부 지능형융합시스템전공
이 석 한

이석한의 공학박사 학위논문을 인준함
2019 년 1 월

위 원 장 이 재 욱 (인)

부위원장 안 정 호 (인)

위 원 김 장 우 (인)

위 원 김 동 준 (인)

위 원 김 남 승 (인)

Abstract

A Stacked Memory Architecture for Improving Performance and Capacity

The advance of DRAM manufacturing technology slows down, whereas the density and performance needs of DRAM continue to increase. This desire has motivated the industry to explore emerging Non-Volatile Memory (e.g., 3D XPoint) and the high-density DRAM (e.g., Managed DRAM Solution). Since such memory technologies increase the density at the cost of longer latency, lower bandwidth, or both, it is essential to use them with fast memory (e.g., conventional DRAM) to which hot pages are transferred at runtime. Nonetheless, we observe that page transfers to fast memory often block memory channels from servicing memory requests from applications for a long period. This in turn significantly increases the high-percentile response time of latency-sensitive applications. In this thesis, we propose a high-density managed DRAM architecture, dubbed 3D-XPath for applications demanding both low latency and high capacity for memory. 3D-XPath DRAM stacks conventional DRAM dies with high-density DRAM dies explored in this thesis and connects these DRAM dies with 3D-XPath. Especially, 3D-XPath allows unused memory channels to service memory requests from applications when primary channels supposed to handle the memory requests

are blocked by page transfers at given moments, considerably increasing the high-percentile response time. This can also improve the throughput of applications frequently copying memory blocks between kernel and user memory spaces. Our evaluation shows that 3D-XPath DRAM decreases high-percentile response time of latency-sensitive applications by ~30% while improving the throughput of an I/O-intensive applications by ~39%, compared with DRAM without 3D-XPath.

Recent computer systems are evolving toward the integration of more CPU cores into a single socket, which require higher memory bandwidth and capacity. Increasing the number of channels per socket is a common solution to the bandwidth demand and to better utilize these increased channels, data bus width is reduced and burst length is increased. However, this longer burst length brings increased DRAM access latency. On the memory capacity side, process scaling has been the answer for decades, but cell capacitance now limits how small a cell could be. 3D stacked memory solves this problem by stacking dies on top of other dies.

We made a key observation in real multicore machine that multiple memory controllers are always not fully utilized on SPEC CPU 2006 rate benchmark. To bring these idle channels into play, we proposed memory channel sharing architecture to boost peak bandwidth of one memory channel and reduce the burst latency on 3D stacked memory. By channel sharing, the total performance on multi-programmed workloads and multi-threaded workloads improved up to respectively 4.3% and 3.6% and the average read

latency reduced up to 8.22% and 10.18%.

Keywords : Memory microarchitecture, Stacked memory, heterogeneous memory, managed DRAM, hot-page swap,
Student Number : 2014-30815

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, there has been an explosive increase in demand for more computational power and data throughput, with this trend projected to continue with time. CPU and GPU, in particular, has evolved to satisfy these demands [63][64][65][66]. To increase the performance of xPUs, decreasing latency while increasing throughput has been main target of manufacturers. In the past, this improvement was achieved by increasing the clock speed of a xPU. However, with the slowdown of silicon process shrinking speed, it has become difficult to increase the clock speed, thereby diminishing the performance gain. To overcome this limit, the focus of high performance computing has instead shifted to parallel computing, or increasing the number of arithmetic logic unit(ALU)s. This change is evident in the increase of CPUs per processor in Intel Xeon and AMD EPYC, and the increase in total ALUs in NVIDIA and AMD's GPU architectures. In the case of Xeon, more compute cores and more ALUs with higher compute bandwidth per core, while increasing cache sizes and reducing latency has been the focus of architectural improvement. In the same regard, GPUs have reduced accuracy while increasing total throughput to improve system performance. However, in accordance with the roofline model [67], while some applications are bound by the compute performance of the processing unit, some application speed is bound

by the speed of the memory as it loads/stores data to the processing units on demand. However, in spite of the improvements in processor architecture, memory performance has remained relatively the same, as the limits in silicon processing hinders performance improvements even at a smaller processes and suffers from insufficient yield [68]. To improve system performance in spite of the relatively slow memory manufacturing process improvements, caches architecture has improved and cache size has increased to improve data reuse, or architecture has changed to increase the number of computation per memory access. For example, Intel Xeon continuously expand the cache capacity per socket. Also, nVidia’s the newest generation GPU has been evolved to have higher buffer capacity and support lower precision calculation mode (INT4, FP16) but it has higher calculation power per access the memory (OP/B) than the previous generation.

However, even with the aforementioned efforts to reduce the impact of memory performance on system performance, as CPUs became more dense and with the advent of applications with little data reuse, such as neural networks [69][70] and datacenter programs[71], system performance has become bounded by memory. To increase memory performance in spite of these limits, memory channels have increased from 4 channels, which has been the norm for quite some time, to 6 channels in the latest Xeon processors, and increasing the number of ranks to increase memory size [73]. GPU based systems have increased the number of data bus channels or implemented GDDRx [28] which has higher data

transfer rate for a single rank of a fixed size.

Increasing the number of channels is also a limited solution. First, the PCB (Print Circuit Board) used today requires more PCB layers for routing, and to secure signal integrity, the layout must become very complex, increasing the cost or making it impossible to implement at all. Second, the number of channels for data transmission between host and memory increases, which significantly adds to the total cost of the system. Third, the number of memory controllers on the host must increase as well, which increases the complexity of the host chip interconnection network and increases the cost as well [76]. Fourth, the IO power consumption increases with respect to the number of increased channels. For these reasons, simply adding a few more channels to increase bandwidth and memory size is no longer plausible.

For decades, DRAM have been connected to host with controller by memory channel and used as a main memory. DRAM was capable providing low latency high bandwidth and large capacity, but with the slowdown of process shrinkage and even with the smaller process, the RC load from increased capacity increases access latency, complicating the timing parameters, nullifying the DRAM parameter gain from the smaller process, and limiting system performance increase.

Introduced as a means to satisfy both the bandwidth and capacity requirements, otherwise impossible by traditional DRAM, stacked memory architecture has been deemed as a suitable solution to the memory wall. Connecting the memory through a semi-conductor

wire connection and not by a PCB channel is the basic principle of stacked memory. By doing so, stacked memory has a channel with a lower RC load and alleviated the aforementioned PCB based channel problems. However, the number of levels that can be stacked is limited, needing another solution to remedy this problem.

To overcome the limitations of DRAM entirely, several different types of memory devices has been researched. PCM (Phase Change Memory), which has been developed under the collaboration of Intel and Micron, is based on 3D-Xpoint to increase the capacity of memory and is supported since the cascade lake system. However, it suffers from long latency and high energy usage, which is why it is used in tandem with DRAM, storing the frequently used data in DRAM and the lesser used data in PCM to improve system performance. The relatively poor performance and expensive price does hinder its usability in actual deployment.

In this dissertation, we propose a new memory architecture to overcome the diminishing DRAM performance improvement and DRAM capacity limit. We aim to improve memory performance by proposing a new memory architecture based on DRAM and not use a different type of memory. We explain the necessary components in implementing the system and analyze the performance improvement when the proposed memory is integrated to a system.

1.1 3D-XPath: High-Density Managed DRAM Architecture with Cost-effective Alternative Paths for Memory Transactions

The demand for larger main memory capacity keeps increasing. However, as DRAM technology scaling slows down, so does the capacity scaling of a Dual-Inline Memory Module (DIMM) consisting of DRAM packages. This in turn constrains the capacity of main memory systems, because the maximum number of DIMMs per memory channel (or simply "channel") is limited to a small number, and it decreases with higher signaling rates. In such a case, it becomes hard to further increase the capacity without sacrificing either bandwidth or latency. Although buffered DIMMs such as Registered DIMMs [27] and Load-Reduced DIMMs [26] allow more DIMMs per channel without sacrificing the bandwidth, they increase the latency.

To improve capacity per DRAM package, we may consider stacking DRAM dies. However, as the number of stacked dies increases, (1) the cost increases super-linearly especially due to drop in yield [20] and (2) the data transfer rate per TSV, which connects these dies, decreases due to worsened skews in setup/hold timings [45]. This challenge has motivated the industry to explore emerging Non-Volatile Memory (NVM) technology such as 3D XPoint or high-density DRAM such as Managed DRAM Solution (MDS) [1]. Nonetheless, both NVM and high-density DRAM increase their capacity at the cost of longer latency, lower bandwidth, or both.

Therefore, it is essential to use such slow memory with fast

memory (such as conventional DRAM) and transfer hot pages to the fast memory to minimize the negative impact of prolonged latency and/or low bandwidth of the slow memory on performance [51, 56]. Analyzing On-Line Data-Intensive (OLDI) applications which often require high capacity and low latency memory, however, we observe that page transfers (or swaps) between fast and slow memories frequently block channels from servicing memory requests of the applications for a long period. This blocking in turn unacceptably increases the high-percentile response time of latency-sensitive OLDI applications. For example, for a main memory system which consists of PCM and DRAM, it takes $\sim 50\mu s$ to swap two 4 KB pages between DRAM and PCM where long latency (e.g., $\sim 1\mu s$ [14]) and low bandwidth (e.g., 50-100 MB/s [14]) of PCM primarily contribute to such a long transfer time. To quantify this effect, we evaluate two configurations stacking DRAM with PCM: (C1) one may block memory requests from the Apache web server and (C2) the other does not during page transfers ("DRAM+NVM with blocking page swaps" and "DRAM+NVM with non-blocking page swaps" in Figure 1). C1 gives 60% longer 95_{th}-percentile response time than the memory system based on only DRAM, whereas C2 experiences just 13% longer 95_{th}-percentile response time. Tackling this challenge, we propose a high-density managed DRAM architecture, 3D-XPath, for applications that require both high capacity and low latency for memory. Specifically, advocating the use of high-density DRAM with conventional DRAM, we first discuss challenges in DRAM

technology scaling and then explore design space of high-density DRAM for the first time. Second, we propose 3D-XPath, which provides alternative paths for the following two purposes: (P1) efficient transfers of pages between dies; and (P2) prompt services of memory requests from applications ("memory requests" hereafter) when primary channels for the memory requests are already transferring pages. To cost-effectively offer 3D-XPath, we exploit the following two observations from our experiment and in-depth analysis of industry 3D-stacked DRAM [32]. (O1) One or more channels are often unused for a certain period. (O2) A set of I/O TSVs constituting each channel is physically connected to the I/O connection points of all stacked dies, and it can be electrically connected to any stacked die at a time by controlling tristate buffers with decoder logic. In a 3D-stacked memory package with multiple channels, O1 and O2 allow a memory controller to dynamically establish an alternative channel for a memory request to one rank although its primary channel, which is shared with other ranks, is used by a page transfer to other ranks. In this thesis, we evaluate 3D-XPath for memory which stacks conventional and high-density DRAM dies and uses the conventional DRAM as hardware-managed cache. Nonetheless, 3D-XPath is also applicable to memory which (1) stacks any heterogeneous memory dies including DRAM and NVM and/or (2) uses conventional DRAM as a hardware-managed cache with large cache lines. Lastly, 3D-XPath can greatly improve performance of I/O-intensive applications that frequently need to copy memory blocks between

kernel and user memory spaces, as 3D-XPath can efficiently support in-memory copy operations.

To evaluate the effectiveness of 3D-XPath, we model a heterogeneous 3D-stacked memory system with two conventional and six high-density DRAM dies. As our baseline system, we use a 16-core chip-multiprocessor system with the heterogeneous 3D-stacked memory system. Through system-level simulations we compare heterogeneous memory systems adopting our proposed techniques with the baseline system. Our evaluation shows that heterogeneous 3D-stacked DRAM with 3D-XPath reduces 95th-percentile response time of OLDI applications by ~30% while improving throughput of an I/O-intensive application by ~39%. Lastly, heterogeneous 3Dstacked DRAM can provide higher capacity than homogeneous 3D-stacked DRAM with only conventional DRAM for the same number of dies, but it still gives worse 95th-percentile response time even for applications demanding high memory capacity unless it adopts 3D-XPath, according to our evaluation.

1.2 Boosting Bandwidth – Dynamic Channel Sharing on 3D Stacked Memory

Modern multicore computing systems are evolving toward a larger number of cores per socket, increasing the demand for higher memory bandwidth [1], [2], [3]. In order to meet this bandwidth demand, the number of memory controllers per socket is increasing. Despite the increase in channels and memory controllers, the data bus width of a single channel has remained the same for several reasons. First, it is to reduce the bottleneck caused by narrow command bus. Assuming that a multicore systems memory access request is performed through a single channel and it has the same transfer size and bandwidth as multi-channel, the data bus utilization per memory access command is lower than the multiple channel case, due to wider data bus width. As a result, the data bus cannot be fully utilized, hindered by the limited command bus bandwidth. Second, multiple memory channels can bring higher DRAM parallelism than the single MC system. If the DRAM channel has the same constraints (i.e., the number of components, banks, and ranks) in the both multiple and single channel system, the total number of accessible DRAM banks in the system is proportional to the number of channels. In order words, when we use a single channel, the performance of the system can be limited by the smaller number of simultaneously accessible banks (Bank level parallelism) [29]. For these reasons, the system is composed of multiple channels that can independently transmit data and the

channel data bus width is maintained. In order to transfer data that is wider than the channel data bus width in one request, it uses burst transfers which divides transmitted or received data several times and transmit it serially. This function will be further enhanced with the upcoming DDR5 standard, which will reduce the data bus width per channel (64 to 32 bits) and increase the length of the burst transfer (8 to 16 bursts) than the existing DDR4. These increased burst transfers help to achieve higher data bus utilization, but as a trade-off, the memory controller cannot use the channel for other purposes during the multi-cycle data transfer. As the number of cores is further increased, not only higher bandwidth but also larger memory capacity is required. Moore's law has improved the manufacturing process and increased capacity per area. However, in recent years, process improvement slows down and DRAM capacity increase has been limited. The main cause of this limit is capacitance of DRAM cell because DRAM needs to continuously refresh its data to retain its data. Data loss is caused by charge leakage and as such, if the cells capacitance is reduced as the cells area is reduced, memory refresh must be performed more frequently. These additional refreshes block normal memory transfer command and causes performance degradation [21]. As a result, in order to maintain DRAM performance, the capacitance of each cell cannot be greatly changed. Therefore, it is difficult to radically increase memory density. To overcome this memory density limit, a 3D stacked memory which piles up a memory die on top of the host or another memory die is proposed to increase the

capacity while maintaining the cell area and each stack is connected by through Silicon Via (TSV). Although the cost is increased by stacking the silicon, the capacity problem is solved by increasing the number of stacked DRAM dies. These TSV-based 3D stacked memories are used in Wide IO [18], DDR4 and DDR5 3DS-DRAM [17], HMC [14] and HBM [19]. In particular, Wide IO, HMC, and HBM can use multiple channels per DRAM die. These 3D stacked memories are widely studied and used in the server [5], mobile [9], and graphics [20]. In a real Intel Xeon e5 server system with multi-core and multi-memory channel configuration, we made a key observation that the memory bandwidth is not fully utilized and saturates even if the rate (how many cores are used at a time) is increased on SPEC CPU2006 benchmark. We also found idle channels during the burst transfer of a specific memory channel for multi-programmed and multithreaded workloads through simulations performed in the same condition as the Section 2.5.1. When using a single channel, the probability that another channel will not be used during burst latency (tBL) is as shown in Figure 2. We observe that at least 73% of total memory accesses have at least one unused memory channel even in the relatively high memory data bus utilization case of mix-high and fft which are described in Section 2.5.2. In this thesis, we propose a channel sharing scheme on a 3D stacked memory to compensate for the burst transfer latency by utilizing empty memory channels in server multimemory channel system. We modify the memory and the memory controller architecture and add communication channel to

the the memory channel. By doing so, we are able to transmit data through memory channels that otherwise would have been left idle. Channel sharing increases data bus utilization while mitigating the trade-off between using a narrow data bus and increasing the burst transfer latency. Thus, this design improves the performance of the system by reducing the DRAM access latency and instantaneously increasing the peak channel bandwidth.

1.3 Research contribution

We propose a large capacity DRAM structure using improved silicon process and a stacked asymmetric memory structure which is denser than a typical DRAM, a managed DRAM structure that utilized the previous structure, and improve upon this by modifying the host side memory controller design.

1. Large capacity memory structure
2. High performance DRAM structure in a stacked enviroment
3. Migration strategy in a heterogeneous memory structure of 1 and 2
4. An alternative path structure to increase migration speed
5. A data swap structure to increase migration speed using 2 and 4
6. In-memory copy structure using 4 and 5
7. Increasing peak bandwidth by bonding the channels in a stacked memory

With the memory structure proposed above, this thesis proposes a DRAM system architecture to mitigate the performance and capacity cap from memory wall.

1.4 Outline

This thesis is organized as follows. Chapter 2 describes stacked memory architecture for capacity and performance and measure performance gain for data intensive applications and multi-programmed / multi-threaded applications. Chapter 3 describes the memory and memory controller architecture with bonding memory channels to boost system performance, and observed performance gain for applications. In Chapter 4, we present the conclusion for our works.

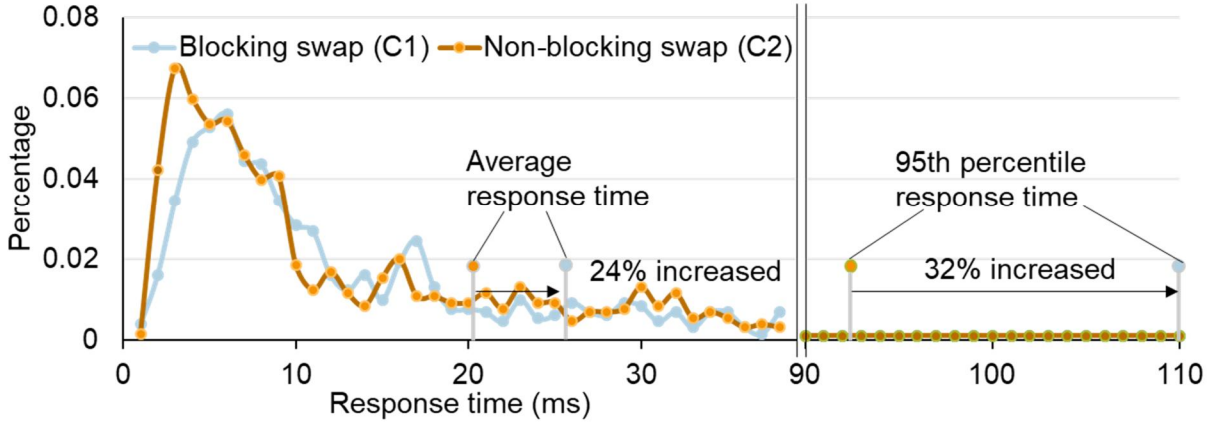


Figure 1 Distributions of Apache response time when main memory systems are built with heterogeneous DRAM where (1) hot-page transfers to faster DRAM block memory requests (C1), and (2) where hot-page transfers do not block memory requests (C2).

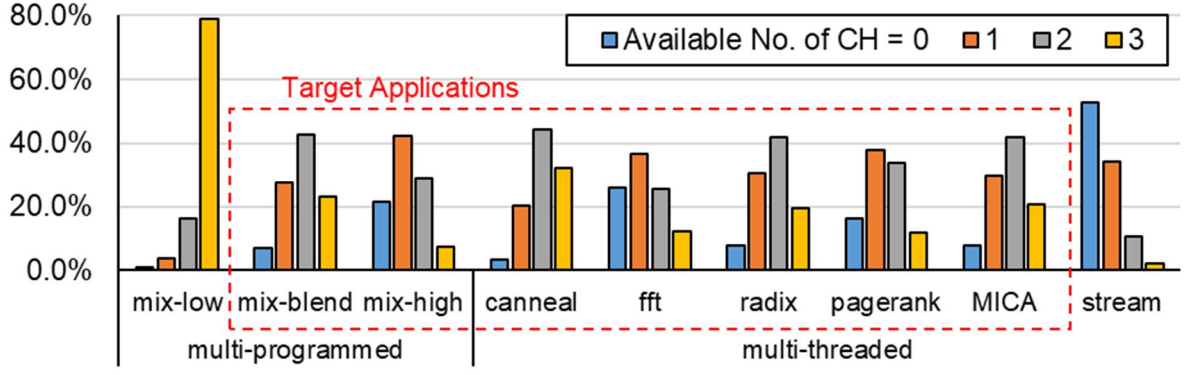


Figure 2 Percentage at which other channels are not used for burst latency when a memory access requested on a memory controller. (This system has 4 memory controllers)

Chapter 2

3D-stacked Heterogeneous Memory Architecture with Cost-effective Extra Block Transfer Paths

2.1 Background

2.1.1 Heterogeneous Main Memory Systems

In a heterogeneous main memory system, DRAM can be deployed as hardware- or software-managed cache, both posing their own challenges. DRAM as hardware-managed cache (e.g., [37, 39]) needs memory space to store tags and decide where to place the tags. Furthermore, DRAM used as hardware-managed cache could be slower than DRAM used as main memory (e.g., cache mode in Intel Knights Landing (KNL) [58]), due to the latency penalty of comparing tags for every memory request. Lastly, since DRAM used as hardware-managed cache is not a part of main memory address space, it is less desirable for memory capacity sensitive workloads than DRAM used as main memory. Alternatively, a heterogeneous memory can constitute unified memory space (e.g., a flat mode in KNL [58]), and let the OS or applications explicitly transfer hot pages to DRAM (Figure 3). However, it is still an active research topic to identify hot pages and when to transfer them to DRAM [48, 49, 62]. Lastly, software-managed cache

approaches for heterogeneous memory systems often need to transfer large memory blocks to DRAM to handle transfer pages. This additional challenge prevents channels from servicing memory requests for a prolonged period and thus significantly increases the high-percentile response time of latency-sensitive applications as Figure 4.

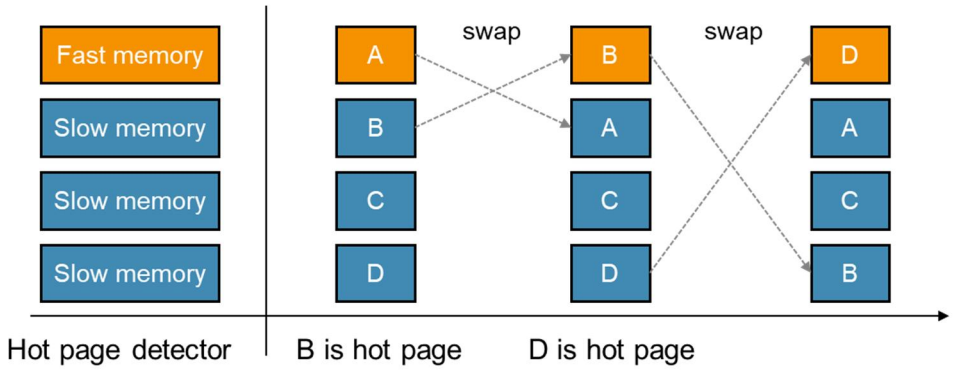


Figure 3 Hot page migration, when page B selected hot page, it is swapped to the fast memory region and A swap out to the original location of the hot page B. At the next epoch, D is chosen as hot page and swaps to the fast memory region again, and page B is swap out to the original location of the hot page D.

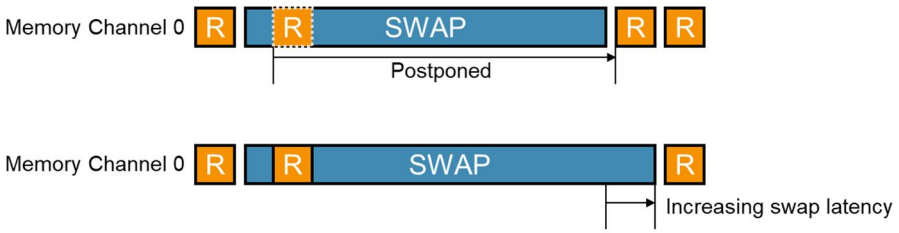


Figure 4 A normal operation during swap blocks the swap operation and generates the additional latency, so swap latency is increased.

2.1.2 Specialized DRAM

Due to many manufacturing challenges, it is becoming harder to improve latency, bandwidth, and capacity of DRAM altogether [38]. However, by not trying to improve every primary aspect of DRAM metrics altogether, it gets easier to make DRAM specialized in certain aspects, such as latency or capacity. As an example, we may consider low-power DRAM as LPDDRx for mobile domain and GDDRx for bandwidth-demanding graphic domain. The latency of DRAM is primarily determined by the time (T1) to deliver command and data signals through global interconnects which traverse DRAM banks; (T2) to sense voltage developed by charge sharing of a DRAM cell and its corresponding bitline (local datapath); and (T3) to precharge the bitlines (BLs) if needed. Son et al. [60] analyzed that the perimeter of a DRAM die determines T1 whereas the BL capacitance determines T2 and T3. As we populate more BL sense amplifiers (BLSAs), we can reduce the latency (e.g., Reduced-Latency DRAM [41]) at the cost of sacrificing the capacity.

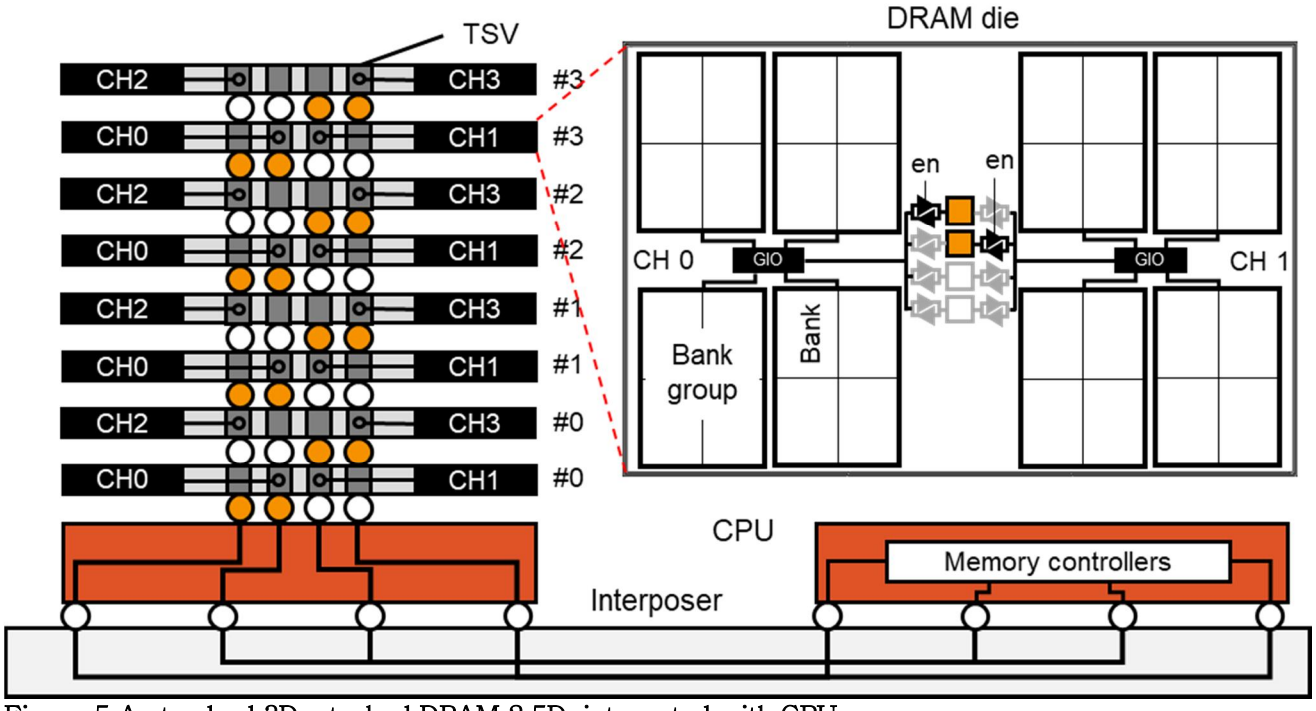


Figure 5 A standard 3D-stacked DRAM 2.5D-integrated with CPU.

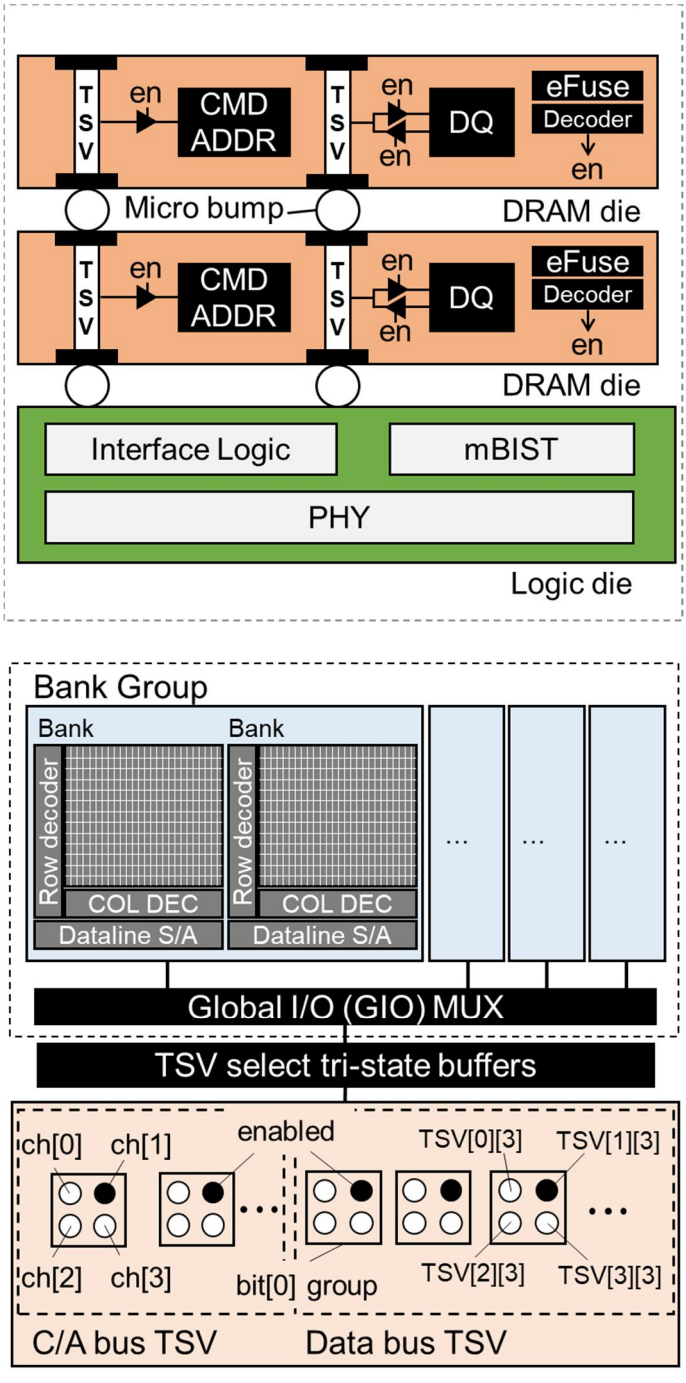


Figure 6 DRAM dies are connected to a logic die through TSVs and micro bumps; a logic die communicates with memory controllers through an interposer.

2.1.3 3D-stacked Memory

Recently, the industry starts to adopt various 3D-stacked DRAM, such as DDR4 3DS [15], Hybrid Memory Cube (HMC) [47], High Bandwidth Memory (HBM) [32], and Multi-Channel DRAM (MCDRAM) [58] for servers and throughput computing. In this work, we take MCDRAM or HBM as a standard 3D-stacked DRAM (Figure 5). Processor units are connected to 3D-stacked DRAM dies through a silicon interposer. A stack consists of several DRAM dies (typically 4 or 8, by default 8 in this thesis) and a logic die. Due to productivity challenges from micro bump mismatch and crack (Figure 8(a)) and the physical limitation from the height of the host silicon, the available number of the stacked die is limited (Figure 8(b)). Also, each datapath TSV (and silicon interposer interconnect) transfers data at 2 Gbps which is limited by internal signal skew and integrity (Figure 8(c)), the transfer rate of HBM2 standard [25]. 512 datapath TSVs per stack lead to 128 GB/s of bandwidth. A stack has multiple channels (4 by default) to utilize this sheer bandwidth efficiently. A DRAM die is connected to one or few channels (2 by default).

We assume that all DRAM dies are fabricated identically to optimize the manufacturing cost like HBM. Then, I/O TSVs constituting four channels are physically connected to the I/O connection points of all stacked DRAM dies. A set of I/O TSVs constituting a channel can be electrically connected to one of the stacked dies by controlling tri-state buffers with decoder logic (Figure 6). The decoder logic can

enable or disable the tri-state buffers, and the manufacturer uses e-fuses to program a Stack ID (SID) into the decoder logic of a stacked DRAM die in the standard 3D-stacked memory. This electrically connects the I/O connection points of a stacked DRAM die to a specific set of I/O TSVs establishing a channel at a manufacturing step. Furthermore, examining the I/O pin layout of the 3D-stacked DRAM, we see that I/O TSVs of all channels associated with the same bit index are closely placed (Figure 7). 3D stacked memory has many channels which contains address word and data word TSVs. Each channel in HBM basically not spread all over the dies but gathered as shown Figure 7. For example, DQ0 of channel 0 is located very near of the DQ0 of channel 1, to prevent complex congestion of metal layer and large data skew variation between data bus of each channel.

TSVs are placed at the center of dies to minimize the worst case topological distance between DRAM cells and TSVs and thus latency [32] [25]. A DRAM die consists of multiple bank groups (8 bank groups per die, 4 bank groups per channel, by default) and each group consists of multiple banks (4 by default). A DRAM bank is organized and operates conventionally; a row to access is first activated at BLSAs (i.e., a row buffer) and then column addresses accompany read/write commands to access data in the row. Within a DRAM die, each bank operates independently except that 1) the number of bank activates within a certain interval is limited due to timing constraints such as tFAW to restrict a surge in current draws which lead to fluctuation in voltage levels, and 2) all banks

mapped to a channel cannot transfer data concurrently due to structural hazard. The bank is structured hierarchically; each bank group has a separate inter-bank dataline to facilitate higher data transfer rate for the transactions heading to different bank groups. A channel and the corresponding bank groups are connected through a multiplexer (mux). The logic die repeats address, command, and data signals between off-stack controllers and DRAM dies.

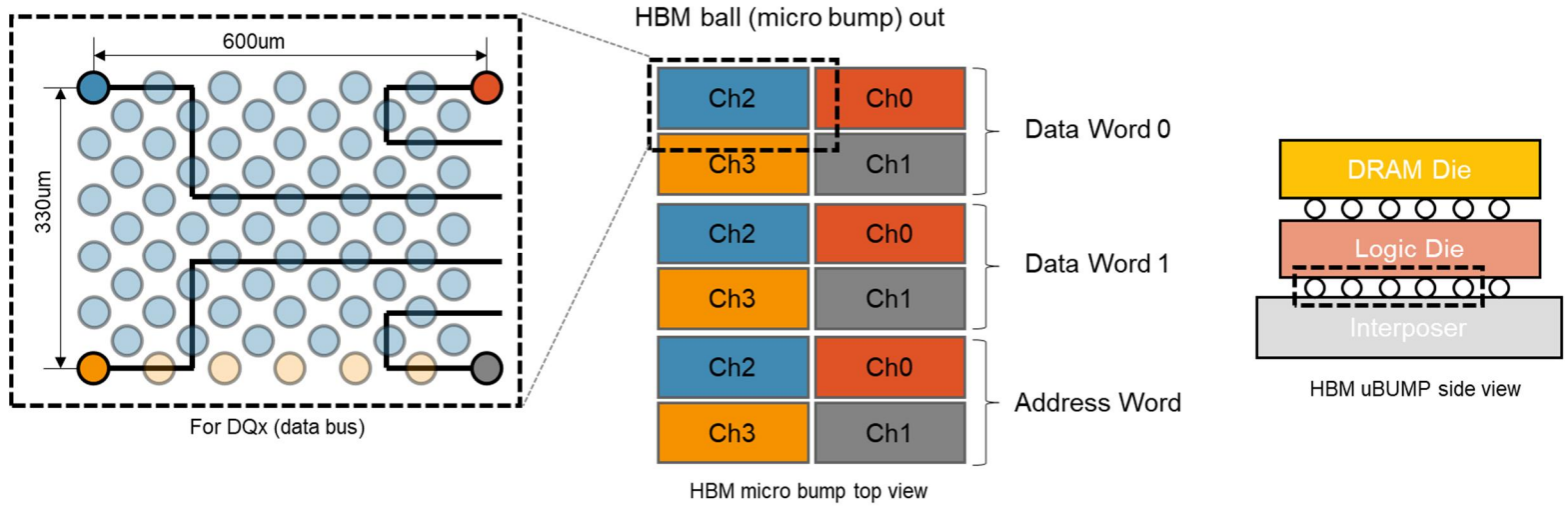
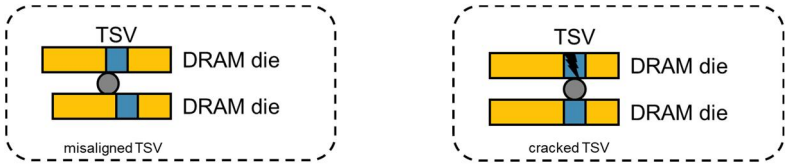
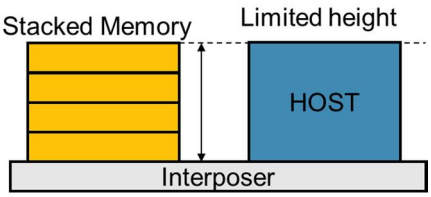


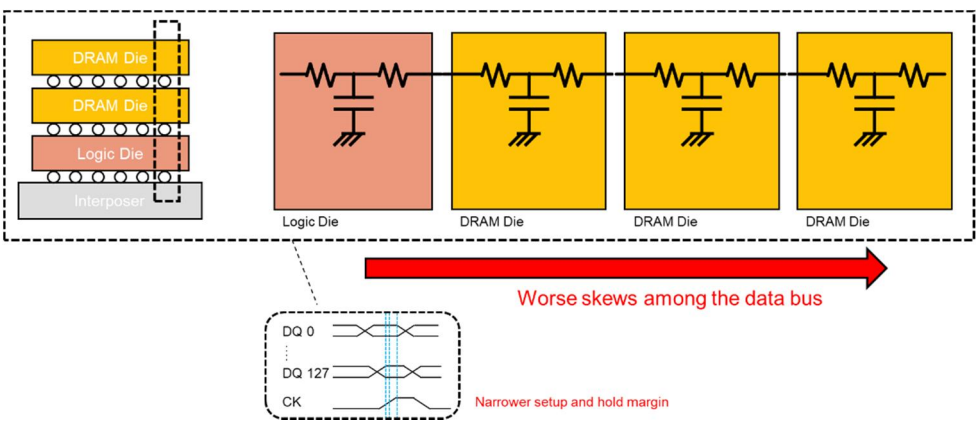
Figure 7 Each data bus pin per channel on an actual 3D stacked memory (HBM) is allocated the similar position. For example, DQ0 for Channel 0~3 are located very near. This characteristic enables communication located very closely, with a small wire latency.



(a) Drop yield by the number of stacks; micro bump mismatch by misaligned bump and cracked TSV



(b) A physical height of stacked memory is limited by the height of the host due to cooling solution



(c) Internal frequency of TSV is hard to increase due to signal integrity, made worse by increasing the number of the stacks

Figure 8 The limitation of 3D stacked memory

2.2 HIGH-DENSITY DRAM ARCHITECTURE

In contrast to low-latency DRAM proposals [41, 55, 60], high-density DRAM, another important class of specialized DRAM designs, is relatively less explored. As latency and bandwidth gap between DRAM and NAND flash increases, the industry introduces solutions to fill this gap, dubbed Storage Class Memory (SCM) [57]. Examples include single-level-cell NAND which focuses on reducing latency (e.g., SanDisk ULLtra DIMMS [10]) and Intel Optane (Phase Change Memory) which exploits 3D XPoint technology with higher density (Figure 9[108]) than normal DRAM but higher latency (Figure 10[109]). A less elaborated but intriguing solution is to specialize DRAM designs for higher density [38]. We identify the key design challenges of high-density DRAM and sketch its plausible design solutions.

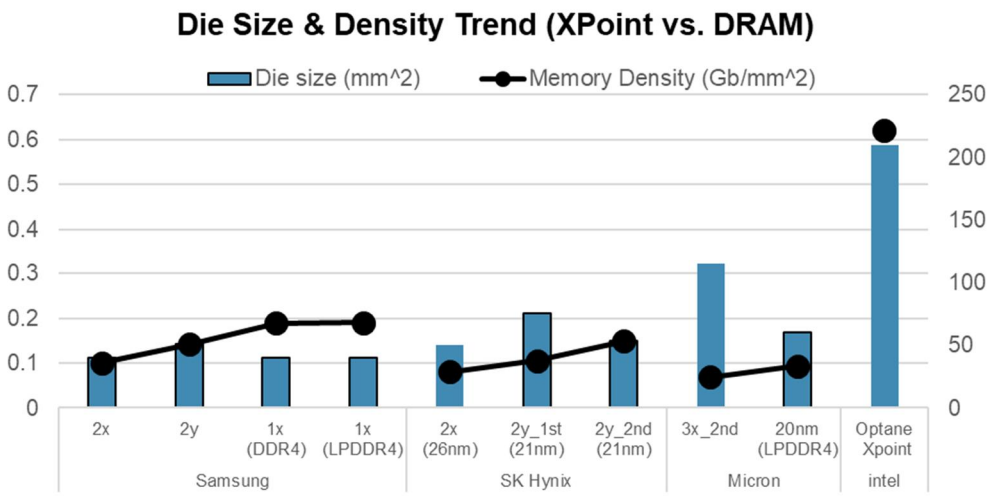


Figure 9 PCM (Phase Change Memory) is widely selected solution. PCM (Intel Optane) cell area per bit is 1/3x – 1/4x of DRAM.

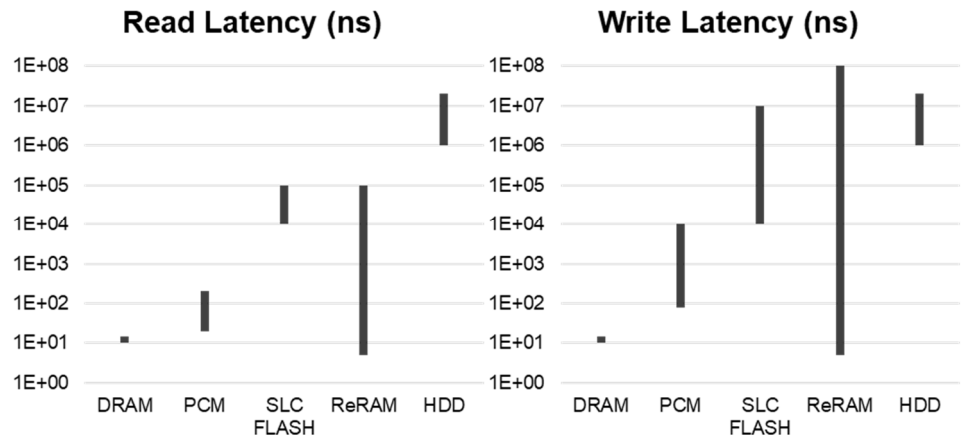


Figure 10 However, PCM operates 2x – 20x slower for read and 8x – 1000x for write than DRAM.

2.2.1 Key Design Challenges

DRAM technology scaling slows down primarily because the DRAM industry is demanded to improve every major aspect of DRAM metrics: latency, bandwidth, and capacity [38]. That is, the DRAM industry can still scale the dimensions of transistors, capacitors, and interconnects (i.e., improving density) for a few more generations like logic and Flash technologies, but it cannot do so while maintaining or improving latency and bandwidth.

DRAM stores data by charges in capacitors, which are detected by sense amplifiers after sharing their charges with bitlines (BLs). A large surface area is needed to increase the capacitance of a DRAM cell, making non-planar designs less viable than NAND Flash where 3D designs are gaining popularity rapidly. Therefore, primary ways to improve storage density of this planar DRAM design is either to increase cell efficiency or to keep scaling down fabrication technology. Increasing cell efficiency (a portion of aggregated cell area over an entire DRAM chip area) has limited potential because a DRAM chip has inevitable circuitry such as inter-bank datalines, off-chip I/O buffers, electrostatic discharge protection, and charge pumps. Moreover, robust sensing requires a certain degree of voltage difference after charge sharing (delta V), limiting the number of cells shared through a BL.

Beyond 20nm process nodes denoted by '1Xnm,' '1Ynm,' and

‘1Znm’ [①]() [11, 38], however, DRAM technology scaling faces three major process challenges. First, fine-pitch metal line and contact resistance increase drastically. The metal resistivity sharply increases below 20nm film thickness because of shrinking in metal volume and the surface scattering effect [46]. This increases the resistance of wordlines (WLs) and BLs. Second, the capacitance of a DRAM cell decreases because cells become smaller and it gets harder to further increase the aspect ratio (or height) of a cell capacitor [38]. Third, the ratio of faulty cells increases as DRAM cell area becomes smaller and process variation is exacerbated.

Significant efforts have made on various aspects of DRAM designs including material (e.g., high-k metal gate to increase transistor speed), fabrication technology (e.g., filling air within spacers to reduce BL capacitance [46]), and cell structuring (e.g., deploying cells like honeycomb to increase cell spacing [46]) to increase DRAM density without sacrificing timing constraints. However, those incur substantial manufacturing costs, need time to be applied stably, or become one-time magic desiring a new solution (another magic) next time. Therefore, without relying on material or process breakthrough, an alternative solution for higher density is to continue DRAM technology scaling while relaxing key timing constraints, which are affected by the aforementioned challenges.

[①] Notations such as 1X and 1Y are widely used terms in DRAM vendors and X, Y, and Z (X > Y > Z) are single-digit numbers, where specific values depend on vendors.

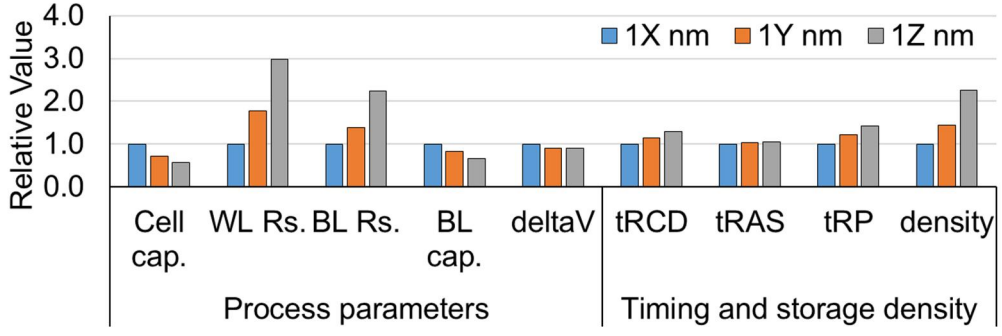


Figure 11 The impact of DRAM technology scaling on process parameters, timing, and density.

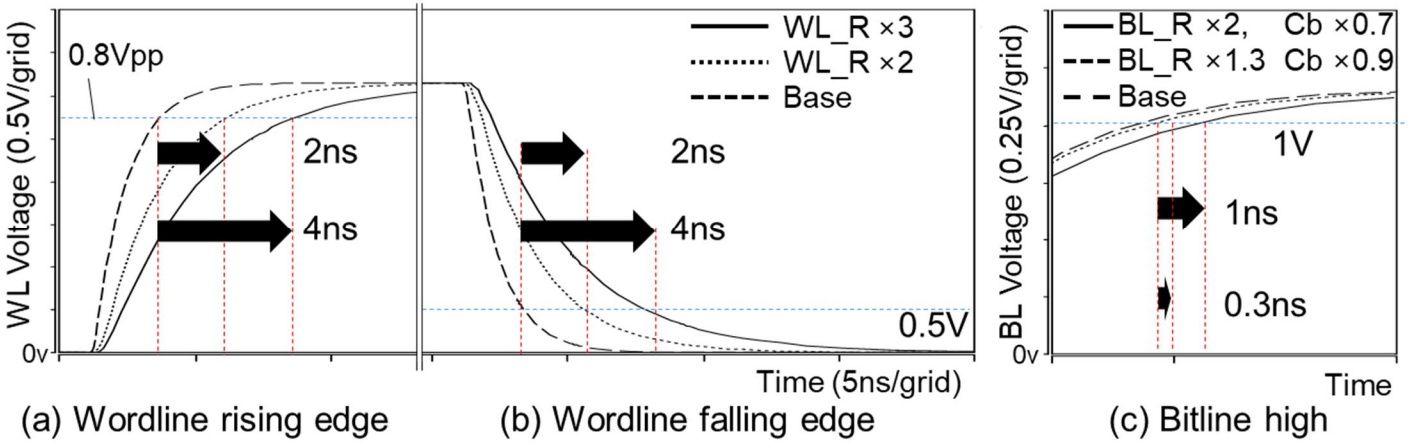


Figure 12 The impact of different resistance and capacitance on timing parameters. When the WL resistance is doubled and tripled, the WL signal is delayed by 2 ns and 4 ns, respectively. When the BL resistance is doubled and its capacitance is decreased by 30%, the BL signal is delayed by 1 ns.

2.2.2 Plausible High-density DRAM Designs

These process challenges can be overcome with the help of relaxing tight timing constraints and architectural support. Increase in resistance of fine-pitch metal lines and contacts slows signal transfer speed and causes timing failure. For example, higher WL resistance delays the delivery of an activation signal to cell transistors located at the far end of a DRAM subarray. That cell transistor turns on slowly; by relaxing tRCD (the minimal interval from an activate to a read/write command on a DRAM bank), a sufficient amount of charge sharing time can be secured. Similarly, the effects of high BL and contact resistance can be suppressed by relaxing key DRAM timing parameters, such as tRP (BL precharge time), tWR (time to write data to DRAM cells), and tRAS (time to destructively read data from a DRAM cell and then restore the data back), because this datapath resistance within a subarray directly affects data restore time and BL precharge delay[②].

Figure 11 projects the impact of primarily scaling the dimensions of transistors, capacitors, and interconnects on major DRAM design parameters as well as timing parameters and storage density, without expecting any significant innovation (magic) in DRAM process technology based on an industry proprietary evaluation setup, whose modeling details are as follows. The cell mat, WL

② A surge in resistance of fine-pitch metal lines and contacts does not affect tCL (read command to first data delay much as global control and datapath use thicker (non-minimal pitch) wires having much lower resistance compared to local WLs and BLs.)

drivers, and sense amplifiers critically affect DRAM timing parameters. The state-of-the-art 6F2 DRAM cell structure has one transistor and one capacitor per crossing point of WL and BL. Total resistance and capacitance values depend on the total number of them. We modeled the critical path by choosing the farthest cell from WL drivers and sense amplifiers of a DRAM mat and conducted SPICE simulation. The voltage-timing diagrams on Figure 12(a) and (b) show SPICE simulation results corresponding to the change of WL resistance, when a WL is developed from Vss (often ground) to Vpp (WL activation voltage) by the WL driver (activation) and from Vpp to Vss (precharge). When the resistance of the WL is doubled and tripled, the signal is delayed by 2 ns and 4 ns, respectively. Figure 12(c) shows simulation result of BL voltage changing from Vss to Vcore (high voltage levels in DRAM cells) according to the change of BL resistance. When the BL resistance is doubled and the BL capacitance is decreased by 30%, the signal is delayed by 1 ns. This shows that the DRAM industry may double the density over two generations (1Xnm to 1Znm) at the cost of increasing key timing parameters such as tRCD, tRAS, and tRP by 28.6% (4 ns due to WL turn-on signal propagation delay), 13.3% (6 ns by BL develop delay and tRCD increment), and 42.8% (WL turn-off signal propagation delay and BL precharge delay).

Reduction in cell capacitance (second challenge) decreases data retention time. More leaky cells have retention time below DRAM refresh window (tREFW). Increase in these leaky and faulty (third challenge) DRAM cells necessitates stronger redundancy. This

problem can be greatly alleviated by recently proposed or applied reliability schemes, such as In-DRAM ECC [11], ArchShield [43], CiDRA [59], XED [44], and Bamboo ECC [33]. These provide cost-effective solutions to common single-bit failures compared to traditional solutions, such as provisioning spare DRAM rows and columns. These schemes typically increase DRAM access latency. For example, In-DRAM ECC, which we assume in this thesis, requires error checking before transmitting data out of DRAM banks, increasing tCL (Table 1).

The aforementioned techniques enable us to get the capacity merit over DRAM taking the conventional evolution path at the cost of lower performance. The pace in reduction of DRAM cell capacitance is faster than that of BL capacitance; and hence more BLSAs should be populated at a given DRAM capacity, increasing area overhead. However, by relaxing timing constraints, BL capacitance can further be reduced; if we make BLs narrower and increase the spacing between BLs, a BL has higher resistance (e.g., deteriorating tWR) but lower capacitance. This helps the voltage difference after charge sharing (delta V) mostly unchanged even after technology scaling (Figure 11) and restricts the rise of area overhead due to populating more BLSAs.

Strong reliability solutions demand additional (parity) DRAM cells (e.g., 6.25% in [11]). Still, storage density improvement (in absolute area) outweighs the costs induced by these techniques. Assuming that these techniques advance the DRAM technology node by two generations compared to the conventional DRAM dies,

high-density DRAM can halve die area for a given capacity as the half-pitch size is reduced by 20% on average per DRAM technology shrink. Either the number of rows, columns, or banks should be increased to deal with more capacity provided by technology scaling. In this thesis, we assume that the number of banks within a bank group is doubled. This will simplify support for page swapping, which will be further elaborated in Section 3.4.2. In summary, we can increase the density by 2.3× at the cost of increasing tRCD, tRAS, and tRP by 29%, 13.3%, 43%, respectively. That is, although we consider the cost of the reliability measures, we can double the capacity per die.

2.3 3D-STACKED DRAM WITH ALTERNATIVE PATHS FOR MEMORY TRANSACTIONS

To build 3D-XPath DRAM, we propose to stack 8 DRAM dies like the standard 3D-stacked DRAM depicted in Section 2.1.3, but we replace top 6DRAMdies with high-density DRAM dies (or Capacity- Optimized DRAM (CapD) dies hereafter) proposed in Section 2.2. This gives roughly 28 GB capacity or 1.75× higher density than the standard 3D-stacked DRAM. Regardless of DRAM types, each DRAM die is connected to two channels and has the same number of bank groups (i.e., four) per channel. A conventional DRAM die (or Performance-Optimized DRAM (PerfD) die hereafter) has 4 banks per bank group whereas a CapD die has 8 banks per bank group. The data bus width of a PerfD die (256 bits) is twice as wide as that of a CapD die. The size of a DRAM row/page is 4 KB for both PerfD and CapD types. The remainder of this section describes how we connect these DRAM dies with 3D-XPath and how we efficiently manage page transfers between CapD and PerfD while servicing memory requests with 3D-XPath.

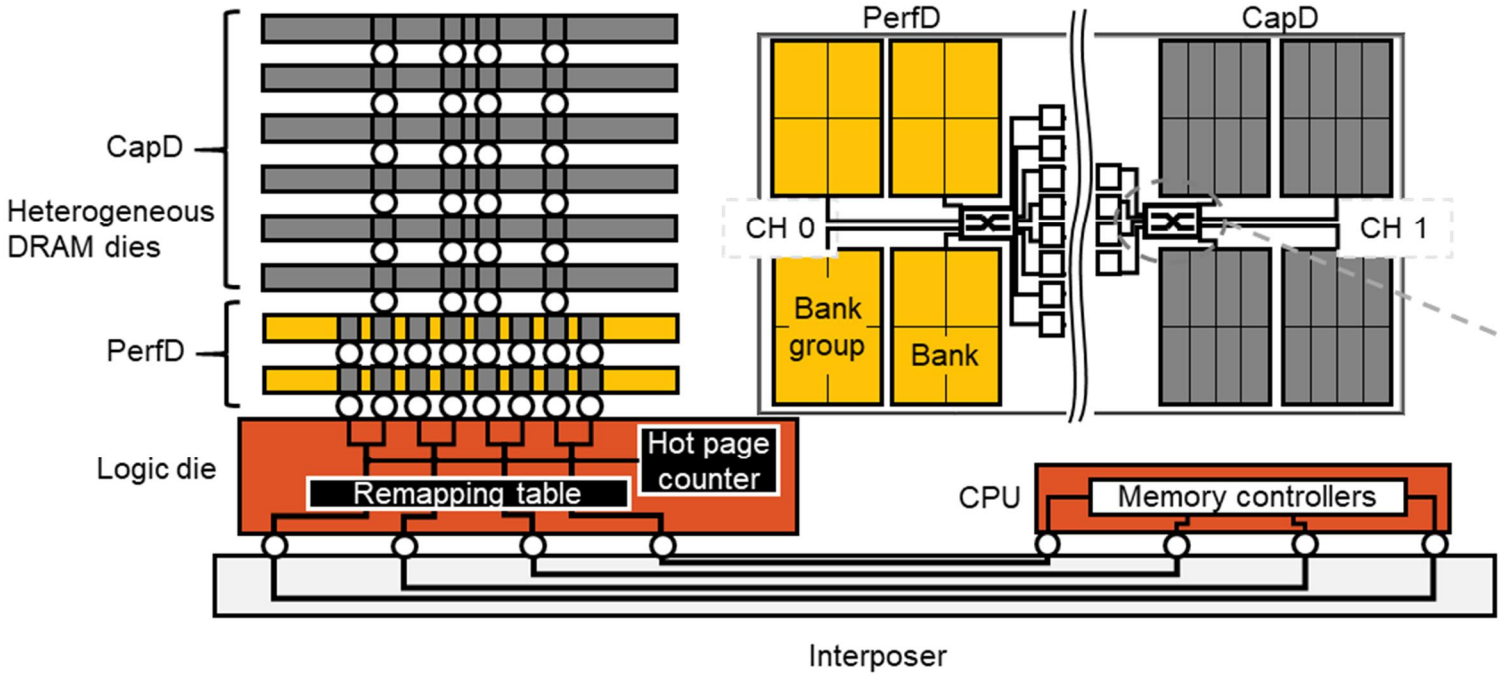


Figure 13 3D-stacked heterogeneous DRAM architecture, which consists of 2 PerfD and 6 CapD dies. PerfD dies have more TSVs for doubled datapath. Crossbar switches to connect TSV channels for 3D-XPath locate at both CapD and PerfD.

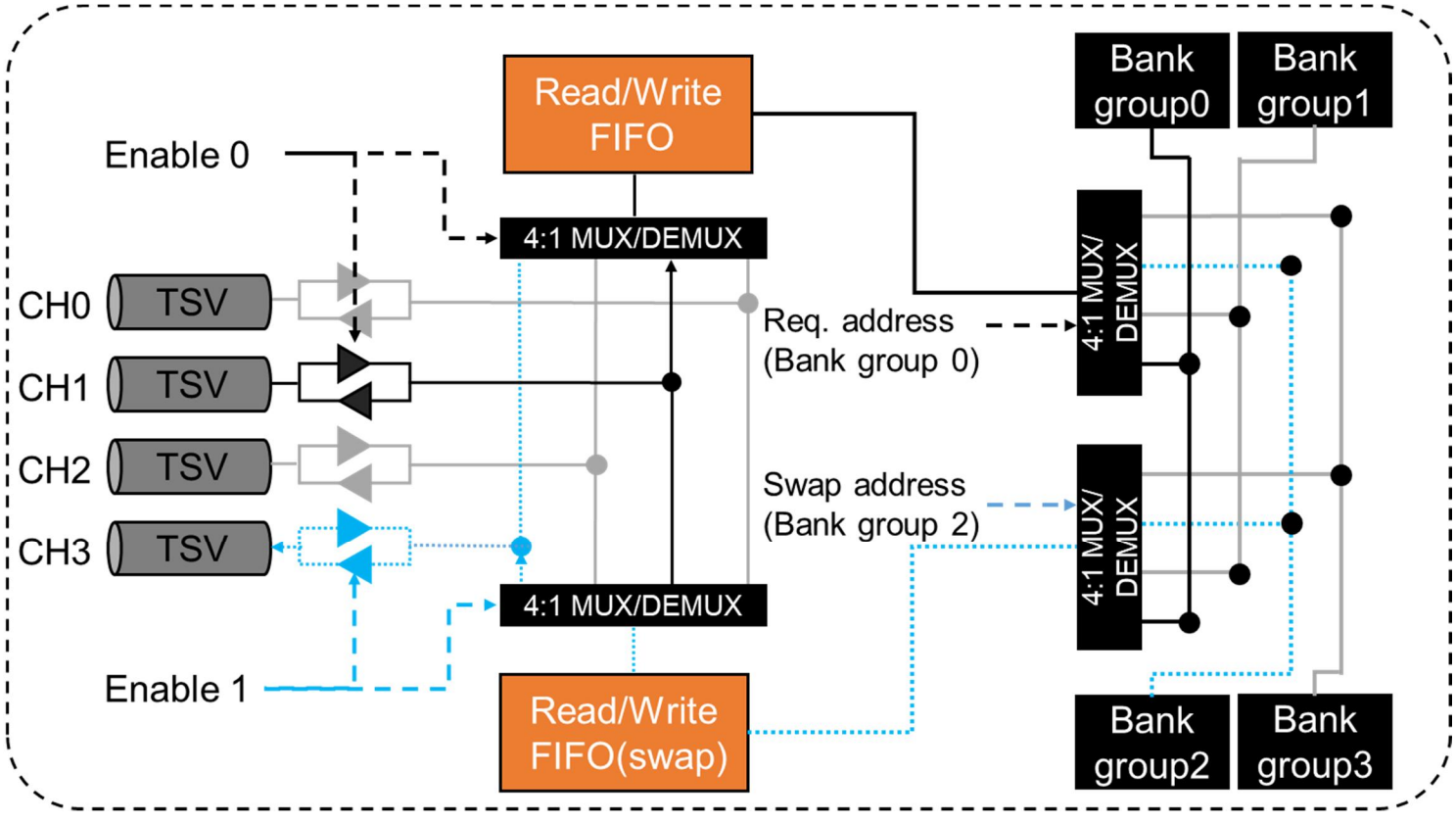


Figure 14 A logic die swaps hot pages between CapD and PerfD. The blue colored blocks are required to implement alternative paths.

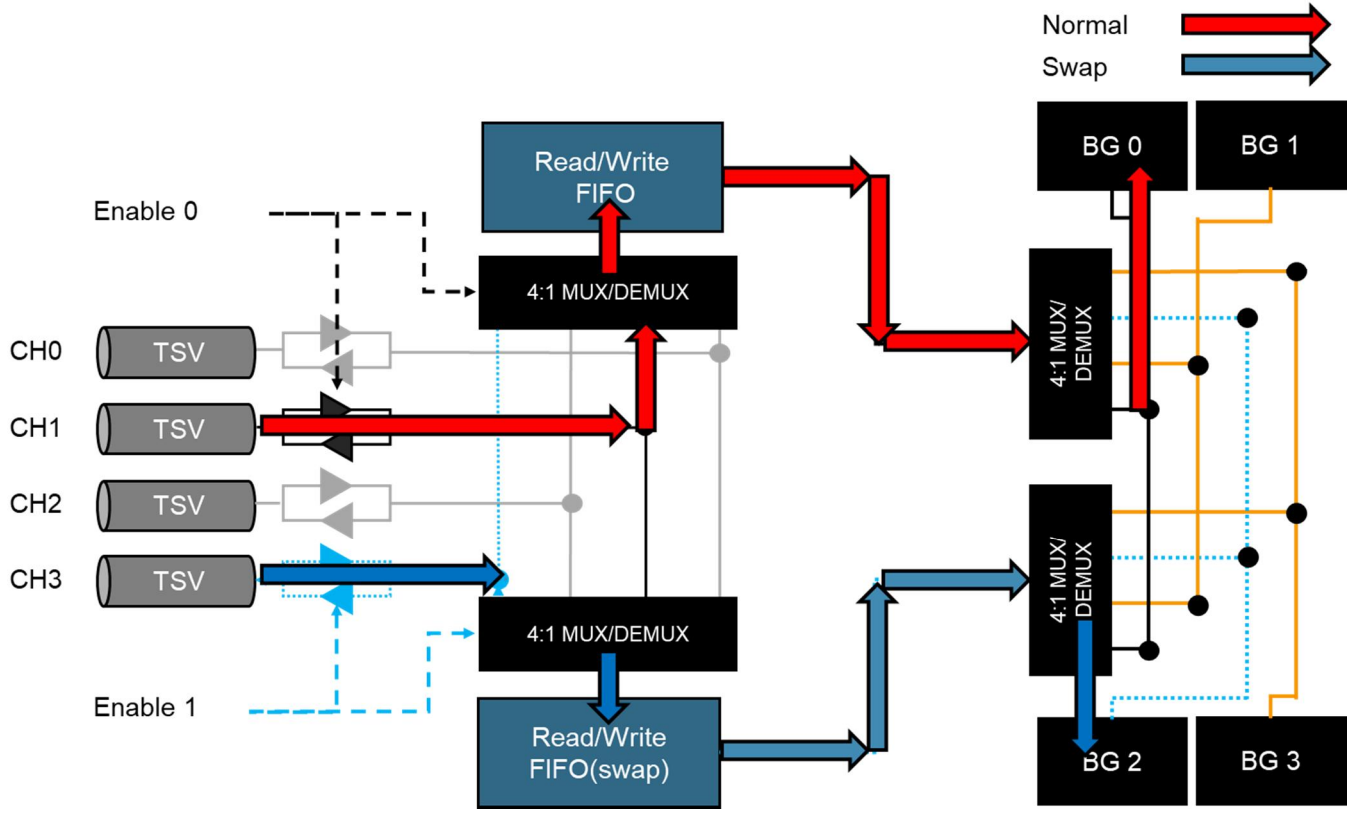


Figure 15 While a swap operation is processing as the blue path, a normal operation is performed on the red path.

2.3.1 3D-XPath Architecture

3D-XPath aims to efficiently service ordinary memory requests from processors while transferring pages between PerfD and CapD, for which we propose 3D-Path and XPath (together 3D-XPath).

3D-Path: A page swap or copy between PerfD and CapD requires many 64-byte (i.e., cache line) memory transactions, blocking memory requests to another bank group of CapD or PerfD involved with swapping/copying pages for a long period. This, in turn, significantly increases the latency of servicing the memory requests, hurting not only the overall system performance but also the high percentile response time of latency-sensitive applications. Tackling this challenge, we leverage two key observations we made from our experiment and in-depth analysis of industry 3D-stacked DRAM. First, one or more channels are often unused for a certain period (Section 2.5.3). Second, all channels are physically connected to all DRAM dies, and a set of I/O TSVs constituting a channel can be electrically connected to any stacked die by controlling tri-state buffers with decoder logic (Section 2.1.3).

From these observations, we propose 3D-Path that diverts memory requests to a less frequently utilized channel at a certain time period, which is feasible with a simple adaptation of the standard 3D-stacked DRAM. Specifically, we can replace the multiplexers, which connect inter-bank datalines (one per bank group) to the channel I/O, with two crossbars (4:1 and 1:4 muxes and demuxes that connect bank groups to channel). Figure 13,14,15 show the

detailed implementation to enable 3D-Path. Each DRAM die has two enable signals per channel for normal memory transaction and migration. For example, a normal access is performed through the predefined TSV channel with en_0 to access bank group 0. A migration should be suspended until the current channel becomes idle without 3DPath, as migration cannot use channel 0 which is already used for the normal access. With 3D-Path, the 3D-Path controller on the logic die controls the extra enable signal (en_1) on each die to send migration data on an idle channel (channel 3 in Figure 15). This allows non-blocking migration operation through the idle channel 3 with en_1 without interrupting the normal memory transaction on channel 1 with en_0. To find an idle channel, we implement a detection logic on the logic die; the logic detects how many cycles will be empty for each channel by a DRAM access command, and the tri-state buffer decoder on the logic die controls the enable signals to use 3D-Path. The crossbars incur little space and timing costs because the I/O TSVs of all channels associated with the same bit index are closely placed (Section 2.1.3, Figure 7). Our estimation using CACTI-3DD and 20 nm DRAM technology information [52] shows that the cost of 3D-Path is 1.5% of a PerfD die. Also, the area cost of the enhanced tri-state buffer controller is less than 0.1% of the logic die.

XPath: It is desirable to increase the transfer rate between a memory controller and PerfD, because effective page swaps lead to more frequent memory requests to PerfD and thus latency surges due to the queuing delay. There are two implementation options to

increase the transfer rate. The first option is to increase the bit rate (operating frequency) of each pin/TSV. This can keep the number of TSVs unchanged, but it is difficult to implement due to worsened skews in setup/hold timings [45] as the bit rate increases. When the number of stacked dies increases, timing skews among datapath interconnects increase further and setup/hold timing margins shrink, limiting the operating frequency of TSVs (Section 2.1.3, Figure 8(c)). The second option is to increase the number of datapath interconnects and TSVs. Specifically, we propose to double the datapath width only for PerfD because CapD is optimized for capacity and more TSVs hurts the density of CapD (Figure 16). As PerfD dies are located closer to a logic die than CapD dies, it is more tractable to handle timing skews among datapath interconnects [45].

Due to wider datapath (i.e., additional DQ[255:128] per channel), the transfers to/from PerfD dies require shorter Burst Length (BL) than those to/from CapD dies. To match the transfer rate between CapD and PerfD, the logic die requires a 512-bit prefetch buffer per channel. This needs more TSVs and thus requires more die space. From prior work [32], we estimate that it requires 2% more die space. To keep up with the doubled bandwidth of PerfD, we also propose to increase the bit rate per pin between the logic die and the host memory controllers from 2 Gbps to 4 Gbps. To support the feasibility of increasing the bit rate through an interposer, we turn to existing I/O implementations: GDDR5X drives 10 to 14 Gbps per pin [28], and LPDDR4X is faster than 4 Gbps per pin on a noisier

PCB channel environment than interposer [29]. We also conducted HSPICE signal

integrity simulation with s-parameter and noise models that were specified in prior work [6], where we observe an enough eye diagram on interposer channels at 4 Gbps.

To further reduce the latency of page swaps, we exploit the doubled datapath and propose XPath, an enhanced page-swap mechanism using wider datapath with two swap buffers per channel on the logic die (Figure 17). Two swap buffers receive pages from a source row (cold page) and a target row (hot page), respectively. CapD uses the first half of the doubled datapath (i.e., DQ[127:0]) while PerfD uses the second half of 128 to 255 bit (i.e., DQ[255:128]) to send two pages to the swap buffers. This transaction requires 128 tCK slots for a 4 KB DRAM page without interrupt. Subsequently, we simultaneously send the pages from the swap buffers to destination PerfD and CapD. That is, a page from CapD goes to PerfD through the first half of the doubled datapath and vice versa through the second half for another 128 tCK slots, reducing the number of 4 KB memory transactions to 2/3.

Lastly, 3D-Path can also use XPath of CH[2] for ordinary memory or page-swap transactions. In this 3D-Path operation, there are two possible exceptions. First, the channel is not physically usable for the same bank group access because they share the inter-bank datapath. Second, a page-swap transaction is paused when there is no unused channel at a certain moment.

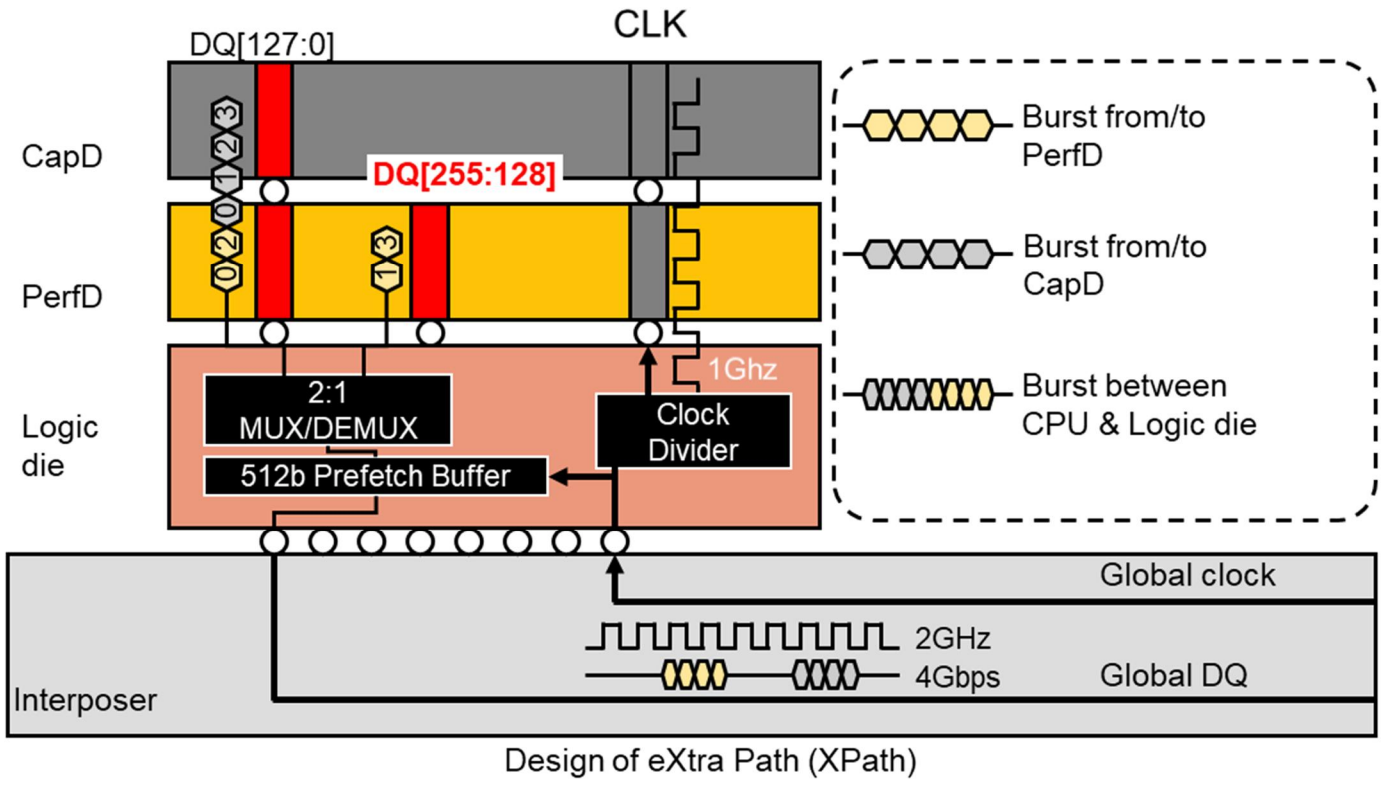


Figure 16 Structure and mechanism for data transfer between 3D-stacked DRAM components with doubled datapath. CapD and PerfD transfer data with 2 Gbps per dataline to the logic die. The logic die can transfer data to a memory controller at 4 Gbps when a PerfD is accessed because a PerfD has twice wider datapath than that between the logic die and the memory controller.

2.3.2 3D-XPath Management

Logic support for page swapping: To swap two pages between PerfD and CapD, we need control logic and support from memory controllers. The control logic can be implemented with simple enhancement of MBIST (Memory Built-In Self Test), which is implemented in recent industry 3D-stacked DRAM [32]. MBIST can generate commands and addresses in a Direct Memory Access (DMA) fashion. The memory controllers also need to know whether a requested page is located at PerfD or CapD. CapD and PerfD have different timing parameters, so we can use a slot-based interface which reserves a slot for slow memory responses on the datapath, similar to prior work [5]. This approach looks for a free slot and reserves the free slot as a backup slot when the page is located at CapD. We also set the transfer granularity of a page-swap transaction to 4 KB, the page size, based on a sensitivity study which is further described in Section 2.5.3.

Page remapping: It is desirable to swap a page in PerfD with any page in CapD, which is similar to a fully associative cache. However, this incurs a huge cost for a page remapping table. Instead, we assume that one page in PerfD can be swapped to one of a few pages in CapD, constituting a page group (similar to Congruence Group in [16]) which can be regarded as a rank. For example, with 2 PerfD and 6 CapD dies, each page group consists of 1 page from a PerfD die and 6 pages from 3 CapD dies (two from each). The pages within a page group are aligned such that they are placed at

the same row address across banks, which significantly reduces the cost of remapping tables.

For a 3D-stacked DRAM with 28 GB of capacity, we need 2.125 MB of memory for the remapping table assuming 3 bits per 4 KB row. The logic die can provide sufficient space to hold this remapping table with SRAM. A more space efficient option is to use a cache and store the entire remapping table in PerfD. For example, a 24 KB remapping table cache achieves a hit rate of 94.1% on average for memory-intensive multi-programmed benchmarks (Section 2.5.2). When a memory request misses the remapping table cache, it brings the corresponding remapping table information from the reserved address space of PerfD. Our estimation using CACTI [13] with 32nm Low STandby Power (LSTP) technology shows that a 16-way 24 KB remapping table cache with 96 B per entry can translate a given page group (rank) address in 0.88 ns, which is less than 1 tCK slot[③] and consumes 0.4 mm2 (less than 1% area of a logic die).

To swap pages, we may follow the sequence described in Row-Clone [53]. That is, the standard 3D-stacked DRAM requires at least three times of row-copy latency (128 tCK slots plus activate/precharge overhead per 4 KB row copy) per page-swap transaction. However, XPath needs only two times of row-copying latency. Besides, we can pause a page-swap transaction by one

[③] tCK is a DRAM clock cycle (1 ns in this study). A DRAM command spends 1 tCK slot.

ordinary memory transaction if a request needs to be sent to another bank group which is not involved with the page-swap transaction. The temporary row buffer can be located at a DRAM or logic die. If it is located at DRAM, a page-sized bulk copy mechanism such as LISA [12], can be applied to reduce page swapping latency between the temporary row buffer and the corresponding BLSAs. However, this requires one more row buffer per bank group of each DRAM. By contrast, when the buffer is located at the logic die, a page swap requires fewer buffers but more transactions. In this thesis, we assume that a temporary row buffer per channel is placed on the logic die.

Interaction with memory requests from applications: During a page-swap transaction, a command for a memory request from applications can interrupt the page-swap transaction. That is, the page-swap transaction is paused since an ordinary memory transaction has a higher priority than a page-swap transaction. However, it is prohibited to precharge or update an opened (activated) row of any bank which is involved with a page swap transaction, because it requires activating the target row again or corrupts values in a page being swapped. To protect against these hazards, if a memory controller sends a command for ordinary memory requests to banks involved with a page-swap transaction, the stacked memory sends an exception signal back to the memory controller through a dedicated pin (similar to AERR and DERR in HBM). Then, the memory controller suspends all commands which head to the bank. After the page-swap transaction is completed,

the memory controller resumes transactions for these memory requests.

Hot-page selection algorithm: In any heterogeneous memory system, it is important to precisely determine which pages are hot because bringing wrong pages to fast memory degrades performance. To determine which pages to swap between PerfD and CapD, we devise a hot page selection algorithm. Specifically, we leverage CHOP-AFC (Caching HOt Pages Adaptive Filter Cache) [30], which was proposed for large DRAM cache, to place hot pages in PerfD without too many page swaps between CapD and PerfD.

CHOP is proposed to choose hot pages based on history counters. Similar to CHOP, our hot-page selection algorithm requires a history counter for each page, and the counter value (set to zero in the beginning) is increased by one when a request is sent to the corresponding page. When the count value of a page exceeds a configured threshold, that page is regarded as a hot page. This hot page will be swapped to PerfD, and the count values of a page group are right-shifted by 1 bit to attenuate the history information (the count values being reduced into half). As mentioned above, our algorithm requires counters for each page and locates them on the logic die of a 3D-stacked memory. For example, a 28 GB memory package which consists of two 2 GB PerfD dies and six 4 GB CapD dies, needs 5.25 MB SRAM storage for a 6-bit counter per 4 KB page. A logic die is large enough to hold that size [32]. The cost can be further reduced by implementing a counter cache in a way similar to populating the remapping table cache as described in

Section 2.3.1. We estimate that the cost of an exemplar 48 KB counter cache located at the logic die is 0.8% based on CACTI [13]. Prior work [30] also proposed an enhanced version of CHOP (CHOP-AFC) to utilize memory bandwidth more efficiently. If the number of memory requests within a certain time period exceeds a threshold, CHOP-AFC transfers hot pages only. Otherwise, it swaps all accessed pages on demand. CHOP-AFC counts the number of memory requests, but it requires a significant amount of logic on the memory controller. By contrast, we monitor the number of pending memory requests in the request queue; such a feature is already implemented in the contemporary memory controller. When the queue size exceeds a given threshold value, the memory controller can send a command or a signal through a dedicated pin to a stacked memory.

High speed in-memory copy: With 3D-XPath, we can also perform any to any in-stack copy with no restriction of source and target addresses. The copy method can utilize the implemented migration hardware and bypass processors. It first activates the source row, saves data in the migration buffer in the logic die, and then copies data to the target row. When the source and destination channels to be copied are different, using 3D-Path implementation, the copy operation is performed without size and address restrictions. The user can control in-memory copy by sending an instruction to the corresponding memory controller. The time taken for copying is two 128 tCK cycles per 4 KB row copy in addition to time for activation of the source and target rows.

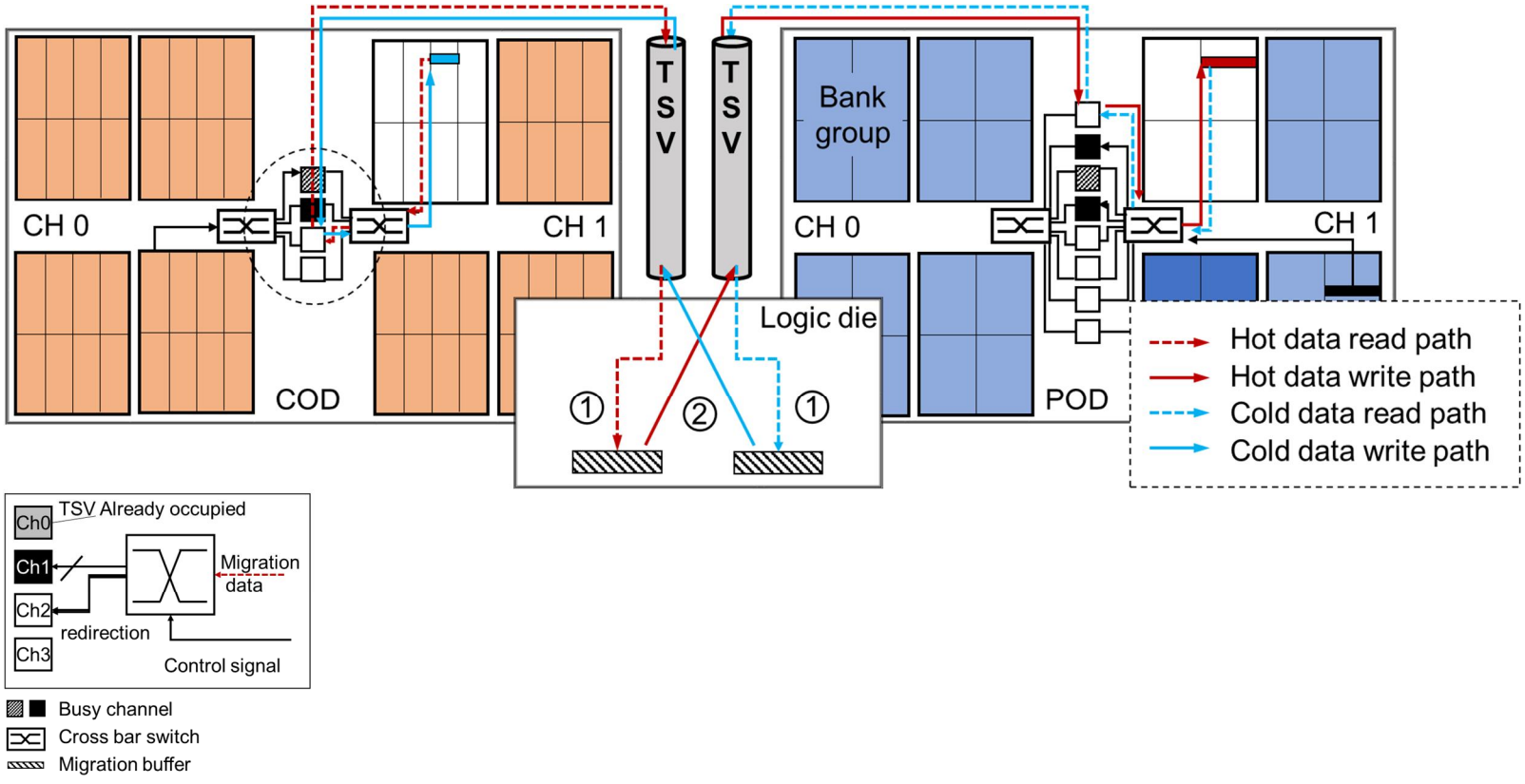


Figure 17 Hot page migration process exploiting 3D-Path and XPath. 3D-Path can redirect a memory transaction from the occupied TSV channel to another free channel. XPath can (1) read data from CapD and PerfD to swap buffers on the logic die simultaneously and (2) write data back to destination dies concurrently by exploiting more TSVs of XPath

2.4 EXPERIMENTAL METHODOLOGY

Description	PerfD	CapD
tRCD	12	18
tRP	13	20
tCL	12	16
DQ size	256	128
Burst Latency	2	4
Capacity Density	1	2

Table 1 Latency of PerfD and CapD

Table 2 SPEC CPU2006 multi-programmed workload groups

Group	SPEC CPU 2006 applications
mix-high0~5	mcf, milc, leslie3d, soplex, GemsFDTD, libquantum, lbm, omnetpp, sphinx3
mix-med	bzip2, gcc, bwaves, zeusmp, gromacs, cactusADM, h264ref, astar, wrf, xalancbmk
mix-low	perlbench, gamess, namd, gobmk, dealII, povray, calculix, hmmer, sjeng, tonto
mix-blend	mcf, bwaves, bzip2, calculix, lbm, milc, cactusADM, wrf, hmmer, soplex, xalancbmk, dealII, leslie3d, libquantum, h264ref, zeusmp

We modeled a chip-multiprocessor system to evaluate the system level impact of the proposed stacked memory architecture. We modified dist-gem5 [4] to simulate full-system configurations with the network subsystem and McSimA+ [3] to support the asymmetric timing parameters for stacked memory, which are enumerated in Table 1. The system has 16 out-of-order cores. Each core operates at 4 GHz, has peak IPC of 4, and is equipped with separate L1 instruction and data caches and a combined L2 cache, all with 64B cache lines. The size and associativity of each

L1 cache and L2 cache are 16 KB and 4, and 512 KB and 16, respectively. A linear hardware prefetcher [31] detects and prefetches streams of consecutive memory accesses. Each memory controller (MC) controls one proposed 3D-stacked memory with four 32 GB/s channels, and has 64 request queue entries. The capacity of our stacked memory is 28 GB. Each channel consists of 4 ranks and each rank consists of 4 bank groups. For PerfD dies each bank group consists of 8 banks, whereas for CapD dies each bank group consists of 16 banks. The memory controller adopts the Parallelism-AwaRe Batch Scheduling (PAR-BS) [42] and adaptive open page management policy [23]. For dist-gem5 simulations, we set the network and storage subsystem parameters to model a 10Gb Network Interface Card and high-performance SATA SSD.

We used two different OLDI applications, Apache and memcached. For Apache, a server system is set with MySQL and Apache2 web server. We prepared Apache with different memory capacity values to show that the increase in memory capacity with CapD affects the performance. We sent enough queries to load the data from the storage into the main memory for these two applications for warm-up; then we made checkpoints and the client sent queries to determine the response time and service rates of the server. When the server receives a query through the Apache2 server, it finds and returns values from the database to the client.

To evaluate the performance of non-OLDI applications, we used four types of multi-programmed (mixed) workloads which consist of SPEC CPU2006 [17] applications executed with reference data

sets. We used Simpoints [54] to extract the most representative simulation points of each SPEC CPU2006 application. Each simulation point consists of 100M instructions. For each multi-programmed workload, a simulation point is assigned to each core, and one or two highest weight points are used per application. We classified each benchmark to one of three groups based on the L2-cache Misses Per Kilo-Instructions (MPKI) [24], each called mix-high, mixmed, and mix-low (Table 2). We populated six multi-programmed workloads from mix-high, called mix-high [0-5]. The other group, called mix-blend, consists of five applications from mix-high, six from mix-med, and five from mix-low. We used SPLASH-2 [61] and PARSEC [9] benchmark suites to evaluate the performance of multi-threaded environment for regions of interest. We used the datasets in [8] for SPLASH-2, and simlarge datasets for PARSEC [9]. We used iperf [2] to measure the performance of the any-to-any in-stack copy proposed in this thesis. iperf is a TCP/UDP-based network bandwidth measurement application that prints network bandwidth of the system. The major network overhead is interrupt processing cost, device driver overhead, checksumming, and buffer copying (the overhead of buffer copying is about 23%) [7]. Exploiting DMA reduces CPU overhead for memory copy, but it still suffers from throughput limitation originating from CPU-side datapath [22]. To show how faster memory copy affects network performance, we compared default C library memory copy, zerocopy using DMA, and 3D-XPath copy.

We obtained the latency value of PerfD and CapD using an industry

proprietary evaluation setup (Table 1). The write recovery delay (tWR) of CapD is not significantly worse than that of default DRAM dies even if CapD employs In-DRAM ECC, as opposed to the prior work [11]. This is because the granularity of data reads and writes to an activated row(i.e., 512b) is larger than the data size of a codeword of In-DRAM ECC (e.g., 128b) because only a single DRAM die is involved in our 3D-stacked memory whereas several DRAM dies participate in data transfers for memory modules in DDRx. The page-swap scheme requires 1.8% more space for remapping table and hot page counter cache on the logic die. In addition, more TSVs and structure for XPath and 3D-Path cost 2% and 1.5% more space, respectively.

2.5 EVALUATION

To evaluate the effectiveness of 3D-XPath in this thesis, we take the 3D-stacked Heterogeneous DRAM consisting of 2 PerfD and 6 CapD dies without 3D-XPath (denoted by He) as our baseline for performance evaluation, unless mentioned otherwise. We test the three other configurations as follows: (1) He with Swapping hot pages (HeS), (2) HeS with XPath (HeSX) and (3) HeS with 3D-XPath (HeS3D-X).

2.5.1 OLDI Workloads

In this evaluation, we set the dataset size of Apache and memcached databases to 4 GB and 2 GB, respectively. We use such memory capacity considering that the full-system simulation time is proportional to the dataset size. However, to appropriately capture the impact of memory capacity on mean and 95th-percentile response times for a given dataset size, we first use a physical machine with larger memory capacity and dataset size. Then we configure our simulation environment such that the simulation gives similar trends in terms of mean and 95th-percentile response times for the chosen dataset size (e.g., 4 GB for Apache). The evaluation of Apache was conducted in two dimensions.

First, we make clients send more Requests Per Second (RPS) than a server can handle, and measure the throughput of a given system in terms of Responded RPS (RRPS), as we change the memory

capacity. In this scenario, RRPS is always smaller than RPS as some requests are not responded. When the dataset size is larger than the memory capacity, page faults (i.e., page swaps between storage and memory) frequently occur, limiting RRPS. For example, Figure 18 shows that 4 GB He gives 78% more RRPS than 2 GB He, which shows the benefit of larger memory capacity. Furthermore, for the same memory capacity, HeS3D-X performs 24% (1 GB), 23% (2 GB), and 17% (4 GB) better than He. Considering the same number of stacked dies per DRAM package, 3D-XPath DRAM can provide 1.75× larger capacity than 3D-stacked DRAM with only PerfD. Although not shown in Figure 18, 14 GB HeS3D-X can provide 11% more RRPS than 8 GB 3D-stacked DRAM with only PerfD, demonstrating the benefit of a larger capacity heterogeneous memory system.

Second, to fairly compare mean and 95th-percentile response times across different configurations, we find the RPS that allows a server with HeS3D-X to respond to all the requests (i.e., RRPS = RPS) and then apply that RPS to all other configurations. Figure 19 shows the mean and 95th-percentile response times for various configurations. Especially, we use uniform and skewed distributions to model the locality of service requests from clients. For the uniform distribution case, the mean and 95th-percentile response times decrease as we increase memory capacity ("apache-uniform" in Figure 19). For example, 8 GB He gives 62% and 66% lower mean and 95th-percentile response times than 4 GB He, respectively. For the skewed distribution case which exhibits the

locality in requested items, 8 GB HeS3D-X offers 29%, 32%, and 23% lower 95th-percentile response time than 8 GB He, HeS, and HeSX, respectively (“apache skewed” in Figure 19).

HeS actually gives 3% longer 95th-percentile response time than He because page-swap transactions block channels from servicing memory requests from Apache. Furthermore, although not shown in Figure 19, compared with 16 GB 3D-stacked DRAM with only PerfD, 28 GB HeS3D-X gives 3% lower 95th-percentile response time, but 28 GB He provides 2% longer 95th-percentile response time. These demonstrate the importance and efficacy of providing larger capacity with 3D-XPath. Figure 19 also shows the mean and 95thpercentile response times of memcached for 4 GB and 8 GB configurations. As memcached stores all data in memory, it does not access the storage device after the warm-up period. For memcached, 8 GB HeS3D-X offers 30%, 32%, and 25% lower 95th-percentile response time than 8 GB He, HeS, and HeSX, respectively.

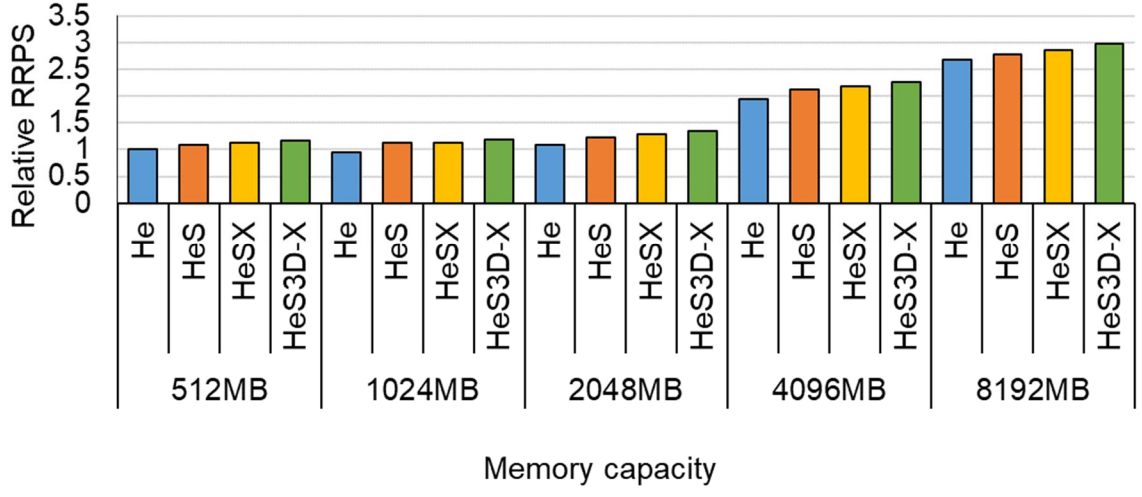


Figure 18 Memory capacity versus performance in terms of Responded Requests Per Second (RRPS) on Apache for “He”terogeneous 3D-stacked DRAM (He), He with “S”wapping pages (HeS), HeS with “X”Path (HeSX), and HeS with “3D-X”Path (HeS3D-X).

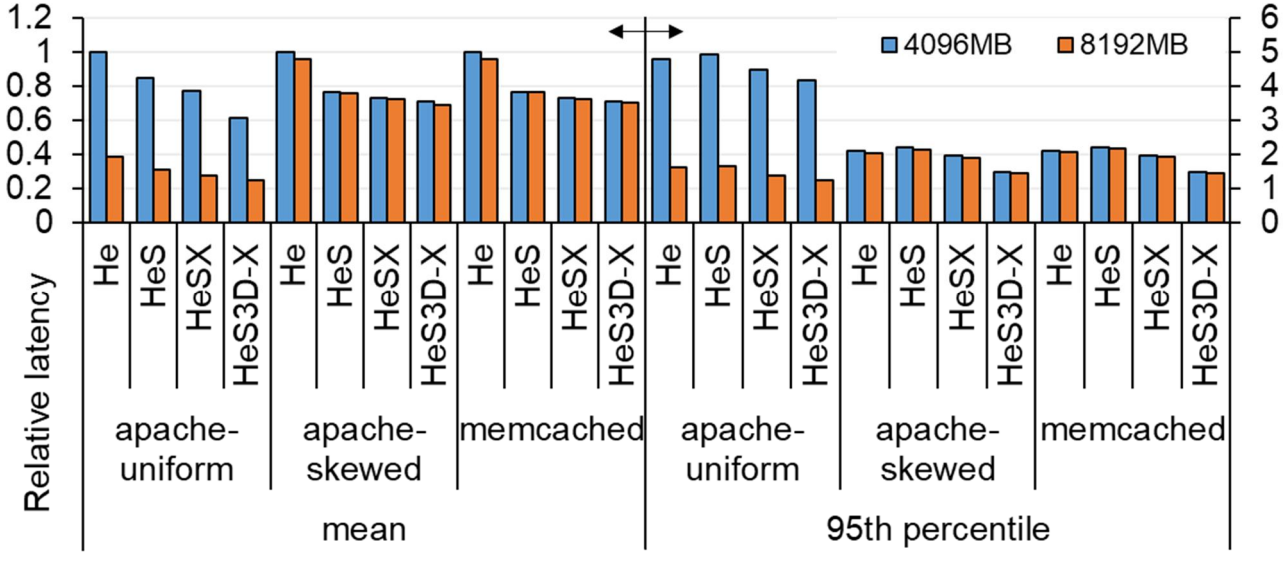


Figure 19 Relative mean and 95t□-percentile response times of Apache and memcached for He, HeS, HeSX, and HeS3D-X, where values are normalized to the mean response time of 4 GB He.

2.5.2 Non-OLDI Workloads

Multi-programmed: For the most memory-intensive multi-programmed workload, mix-high0, HeS3D-X provides 4.9%, 3.2%, 2.0% higher performance and 5.2%, 4.5%, 3.4% lower read latency than He, HeS, and HeSX, respectively. This shows the benefit of 3D-XPath which reduces the negative impact of additional queuing delay imposed by more frequent accesses to PerfD. Evaluating all six mix-high workloads (Figure 20), HeS3D-X provides 2.8%, 1.9%, 1.0% higher performance and 3.4%, 3.3%, 1.6% lower read latency than He, HeS, and HeSX. As expected, mix-low workloads do not benefit from HeS3D-X in terms of performance and read latency because they are less memory-intensive.

Multi-threaded: For memory-intensive radix, HeS, HeSX, and HeS 3D-X give 1.1% 2.6%, and 5.9%, higher performance (Figure 21) than the baseline He, respectively. radix has a high degree of locality in memory accesses, and hence utilizes the lower access latency offered by PerfD dies effectively. For other workloads, HeS3D-X provides up to 2.6% higher performance than He.

Memory-copy bandwidth: To compare memory-copy performance, we ran iperf over various TCP window sizes. We tested three configurations of 1) using standard C library's memory copy (normal), 2) exploiting DMA feature implemented at iperf (DMA), and 3) leveraging the copy instruction proposed at 3D-XPath (3D-X) on Figure 22. With the proposed 3D-XPath copy, we achieve higher network bandwidth regardless of TCP window size (Figure

11). DMA can improve performance by offloading memory copying, but its degree of improvement is still limited. When TCP window size is 416 KB (the default maximum TCP window size of Linux), proposed 3D-X copy provides 39.8% higher network bandwidth than the normal memory copy.

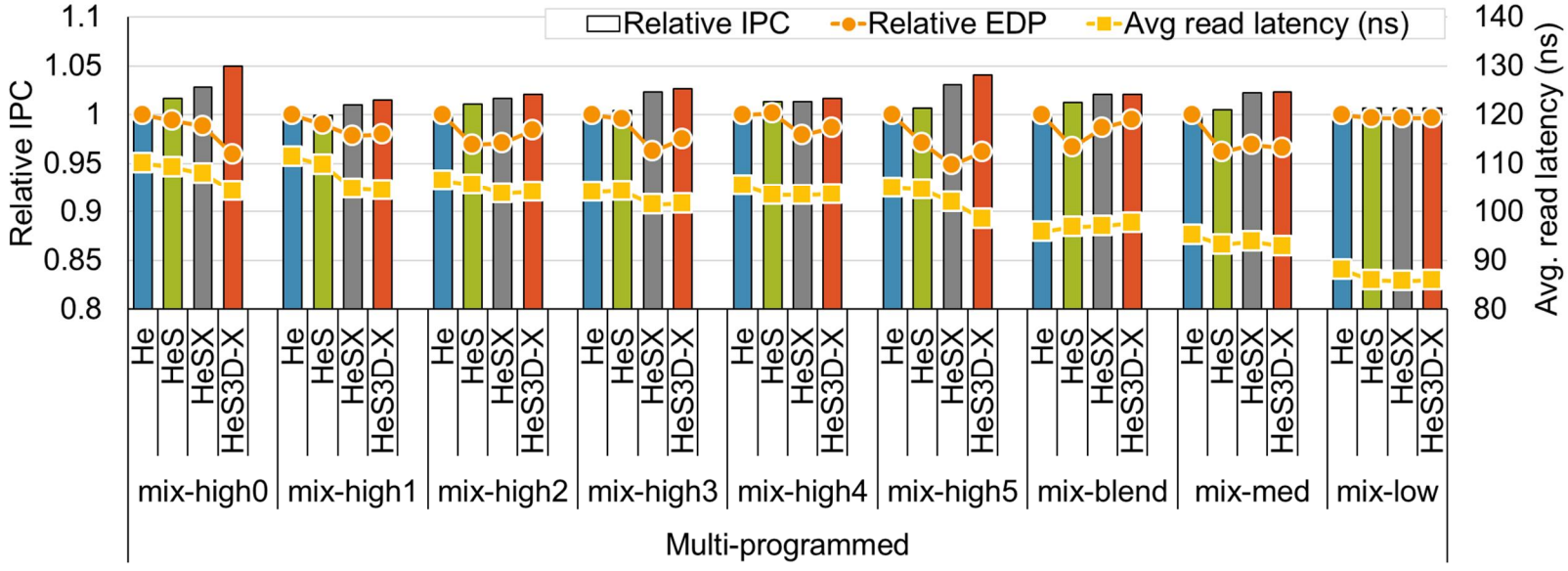


Figure 20 Relative IPC, EDP and average memory latency to handle LLC misses for multi-programmed .

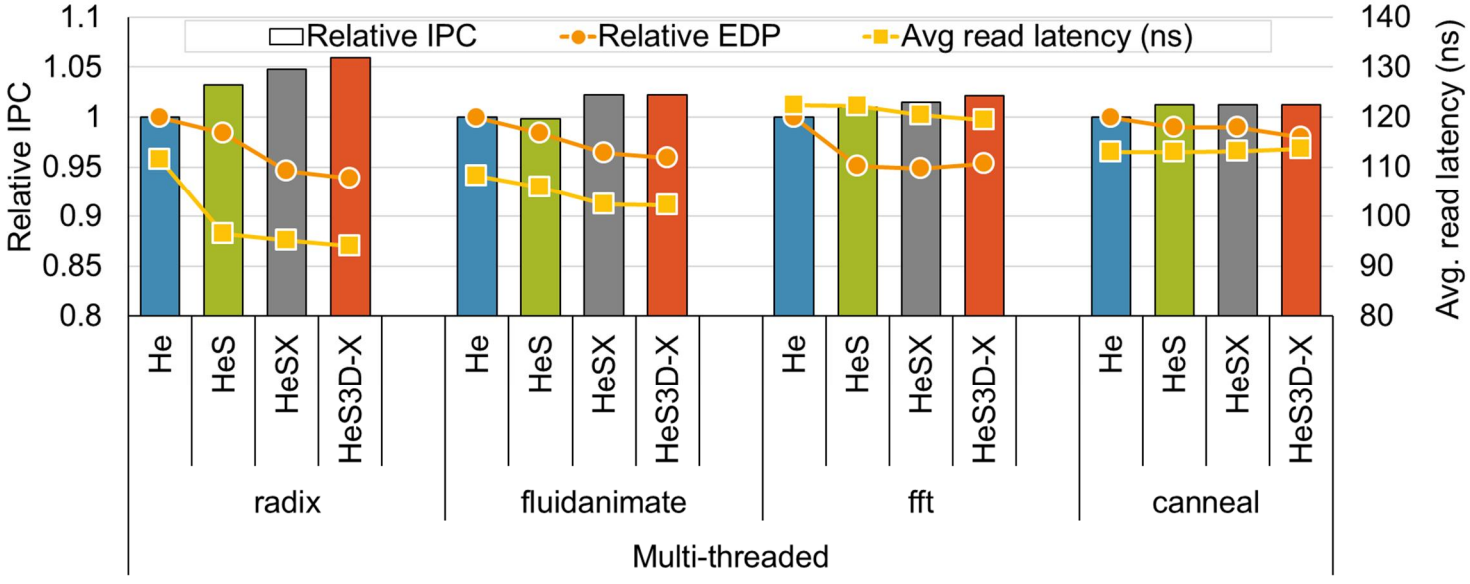


Figure 21 Relative IPC, EDP and average memory latency to handle LLC misses for multi-threaded workloads.

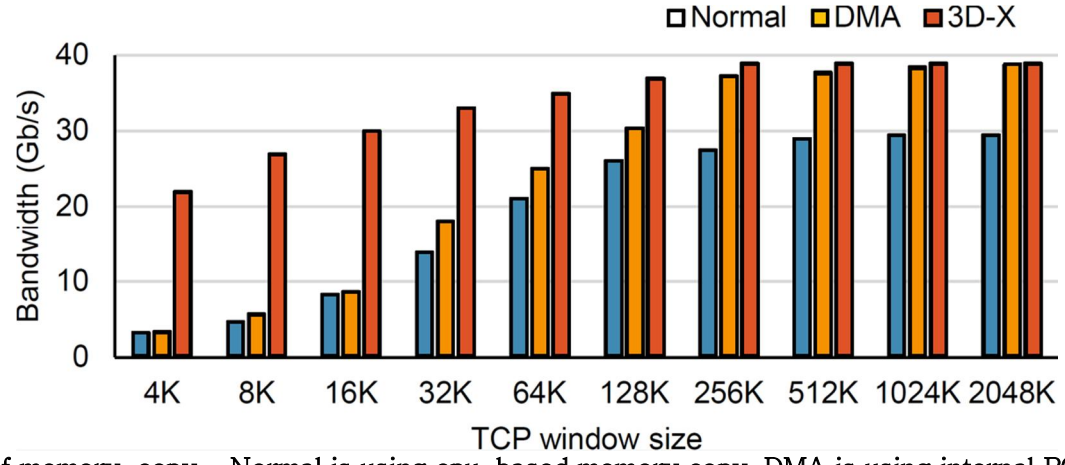


Figure 22 Performance of memory-copy. Normal is using cpu-based memory copy, DMA is using internal PCIe DMA, and 3D-X is using alternative path (3D-X Copy).

2.5.3 Sensitivity Analysis

PerfD hit rates over PerfD and CapD ratios: We change the capacity ratio between CapD and PerfD and evaluate mix-high0 to observe the effectiveness of the hot-page swapping algorithm. Figure 23(a) plots the PerfD hit ratio of He (black bar) which is matched with the capacity ratio between CapD and PerfD. By applying hot-page swaps (HeS), the PerfD hit rate is increased by 8-9%. As 3D-XPath makes page swaps faster, it further improves PerfD hit rates by up to 5%.

Performance sensitivity to page size: We conduct more evaluations to find the rationale behind choosing 4 KB as a row buffer size, which is also used as the unit size of a page swap. Figure 23(b) shows the relative IPC of HeSX and HeS3D-X over various page (row buffer) sizes (the baseline is HeS with 1 KB page on mix-high0). The 4 KB page size performs best because smaller page sizes (e.g., 1 KB and 2 KB) lead to too frequent page swaps, whereas larger page sizes (e.g., 8 KB) take too long per page swap during which ordinary memory transactions to the corresponding bank groups are blocked. Moreover, a smaller page size requires a larger remapping table and more space for hot-page swap counters.

Migration block size: Figure 24 (a) shows how different migration block sizes affect performance. On mix-high0, 512 B leads to the highest IPC, but 4096 B and 1024 B sizes achieve the highest IPC on mix-high1 and mix-high2, respectively. Performance is not much sensitive to migration block size. Thus, similar to page size,

we use 4 KB page size as smaller granularity needs larger remapping tables and hot-page swap counters.

Channel utilization: Evaluating the 3D-stacked DRAM only with PerfD with mix-high0, we observed that multiple channels are not utilized at a certain moment as shown in Figure 24 (a) (an interval of 400μs is presented with the sampling rate of 2.5 MHz). To further support this observation, we monitor all memory transactions and measure the number of cycles for which a given channel stays unoccupied (Figure 23(c)) for the same simulation. At least one channel is empty during 8 cycles for more than 80% of total simulation time. This means that we can efficiently utilize these idle channels by 3D-Path.

Page swapping threshold: Figure 23(d) shows the relative IPC of HeS, HeSX, and HeS3D-X for different page swapping threshold values, which indicates that '63' (6 bits) gives the highest performance. If we apply more than 6 bits for a hot-page counter, the performance improvement would diminish or page swaps occur very infrequently. On the contrary, using smaller threshold values incurs very frequent page swaps and increases time spent for page swaps.

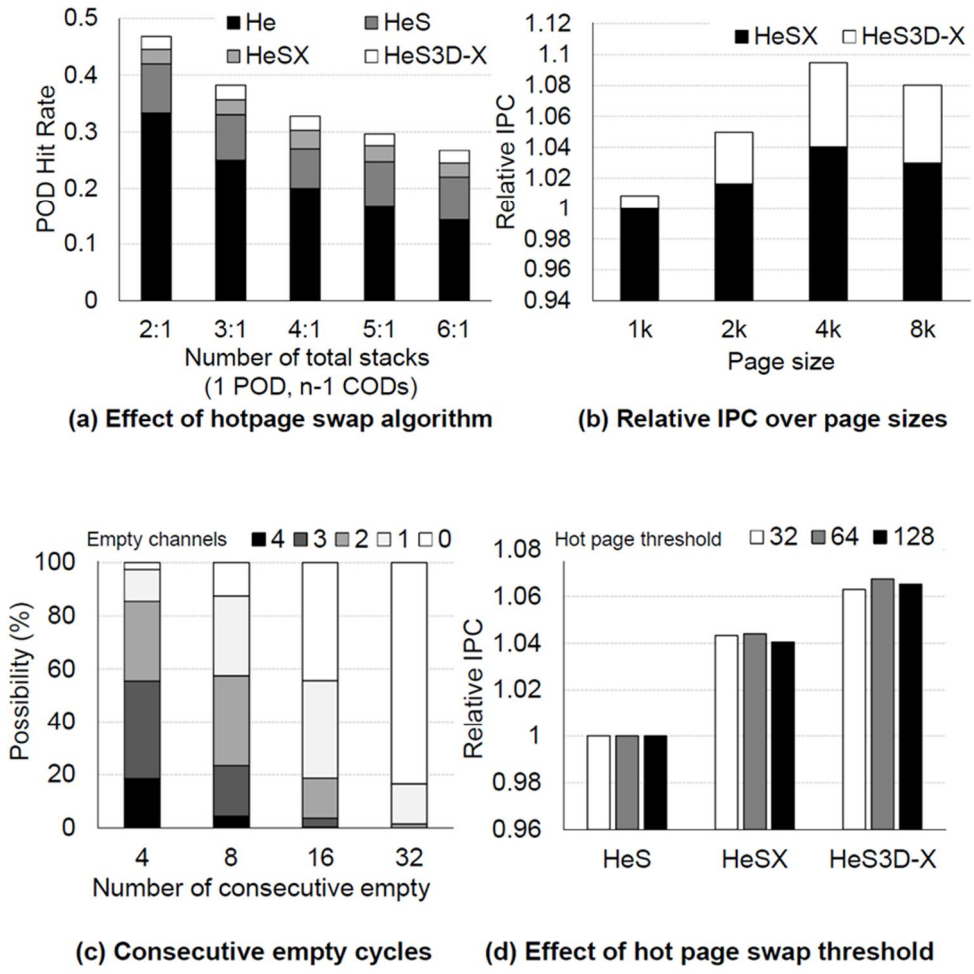
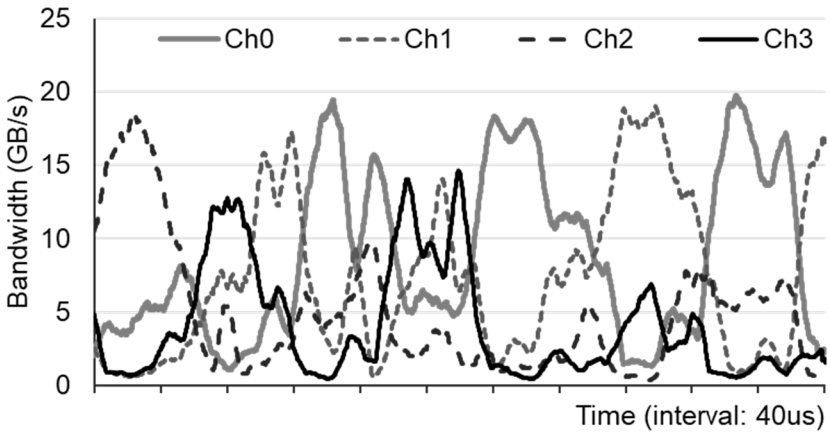
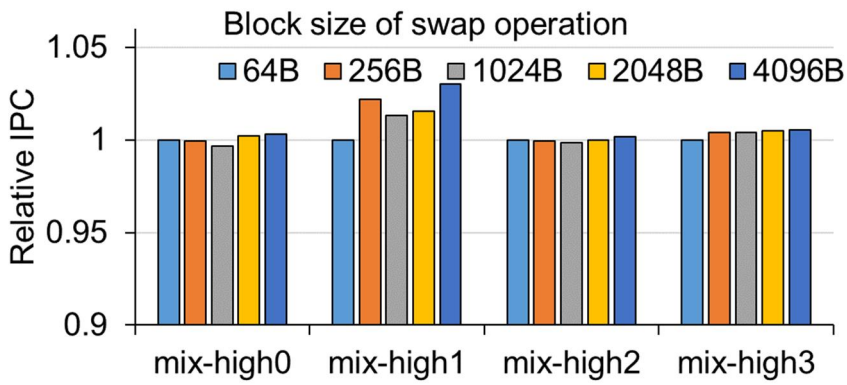


Figure 23 Sensitivity analysis for sweeping each conditions.



(a) Channel utilization by the time on the part of mix-high0



(b) Block size of swap operation with page size 4kB.

Figure 24 Sensitivity analysis for sweeping each conditions.

2.6 RELATED WORK

Enhancing I/O performance of 3D-stacked DRAM: Lee et al. modified the TSV structure on 3D-stacked DRAM for simultaneous multi-die accesses [35]. As opposed to 3D-XPath, they assumed much higher bandwidth between dies and proposed that each die has different I/O bit rates.

Asymmetric DRAM: CHARM changed the aspect ratio of a DRAM mat, a small two-dimensional array of cells, to reduce access latency of DRAM banks depending on physical distance between a bank and I/O [60]. TL-DRAM [36] divides bitlines within a mat into two and provides a way to access a portion of cells closer to sense amplifiers with lower latency. Shin et al. observed that newly refreshed row has more charges and proposed to access them with lower latency [55]. These are orthogonal to 3D-XPath.

NVM+DRAM hybrid memory systems: JEDEC introduced a NVDIMM standard [21], where a DIMM combines DRAM and NVM where NVM preserves data of DRAM (NVDIMM-N) or composes a larger memory space with DRAM (NVDIMM-P). 3D-XPath can provide a high-performance memory-controller-agnostic solution to NVDIMM-P types of memory. PCM has been proposed as a sole replacement of DRAM for main memory [34] or combined with a last-level DRAM cache [40, 48]. They either adjusted row buffer size to alleviate the power impact of PCM writes or populated more row buffers to mitigate the performance penalty of high PCM latency. Dhiman et al. proposed another hybrid system where

DRAM and PCM constitute separate partitions in a single address space [18], and they augmented memory controllers with access map cache and page swapping manager to reduce power consumption. Meza et al. proposed fine-granularity management for DRAM cache [40]. They proposed TIMBER for caching metadata for recently accessed rows in a buffer, with storing tags in memory. They replaced a large SRAM tag memory to small SRAM tag cache.We used a small sized translation SRAM cache to achieve similar benefits.

Page placement in hybrid memory systems: Dong et al. migrate hot pages from an off-chip memory to an on-chip memory [19]. Through modifying on-chip memory controllers, it can manage migration between off-chip and on-chip memory. They focused on how to utilize on-chip and off-chip DRAM with migration, whereas 3D-XPath is between hybrid off-chip memory types. [50] architected a memory controller to observe access patterns and to migrate page frames to PCM to DRAM. They proposed migration policies and augmented memory controllers to monitor access pattern whereas our thesis suggested an alternative path and room for migration on stacked hybrid memory with built-in swap control logic. CAMEO [16], ToR [56], and SILC-FM [51] suggested mechanisms to swap data between fast and slow memory at the granularities of cache block (CAMEO), page (ToR), and in-betweens (SILC-FM). Compared to these proposals, 3D-XPath focused on optimizing the microarchitecture of fast and slow memory being stacked together to provide cost-effective

alternative path between heterogeneous memory dies.

Chapter 3

Boosting bandwidth -Dynamic Channel Sharing on 3D Stacked Memory

3.1 Background: Memory Operations

3.1.1. Memory Controller

In a modern computer system, a read or write request from the host is assigned to a different bus node chosen by a predetermined address mapping via a bus interconnect [91]. Requests to the DRAM memory address region are then transferred to the memory controller. In order to communicate with the memory, the memory controller stacks the requests in its internal queue, and rearrange them according to its policies [104]. After that, the memory controller converts them to the external interface format such as DDRx [93]. These memory controllers are designed to be physically adjacent to one another within a chip. In particular, for Intel Xeon broadwellx and earlier, 2 memory controllers are next to each other, and for Intel Xeon SkylakeX, 3 memory controllers are adjacent. Those memory controllers are assigned to one bus interconnect node (home agent). For AMD's EPYC, up to eight memory controllers can be connected per socket, and each socket is made up of four multi-chip modules. In addition, each module can locally use two memory controllers which are located very close to each other and the other memory controllers work as remote

memory controllers.

3.1.2 DRAM column access sequence

The read or write commands requested by a host on a channel are generally performed in units of cache line size. In a typical computing system, the cache line size is 64 B. Since the area and performance overhead from wider bus is larger when 64 B are transmitted at once with off-chip interface, the data bus width is remained narrow. For example, DDR3 and DDR4 are configured with 8 B per channel [93] and DDR5 is configured with 4 B per channel. The memory controller sends a read command to the command address bus and awaits data transfer for column latency (tCL), the time required to transfer data through DRAM internal datapath. After tCL, data is transmitted from the memory through the data bus, and in the case of a write, the memory controller transmits data after column write latency (tCWL). In both cases, data is transferred for burst latency (tBL), and stored in a temporal storage called the prefetch buffer. The consecutive transfer between prefetch buffer and memory controller is called burst transfer. The length of this transfer is called burst length (BL). For example, BL of DDR3 and DDR4 is 8 and DDR5 performs 16 burst transfers. For read, after this burst transfer, the data buffer is filled in the MC and transferred to the requested cache through the on-chip network and for write, the write FIFO in prefetch buffer is filled and reflect to DRAM cells for write recovery time. As a result, the FILO (First In Last Out) latency required to process one column

access command, which is the time the request actually remains in the queue on the memory controller side, becomes tCL + tBL.

3.2 Related Work

Chen et al. [84] proposed that a switch is mounted on the power pin and the rank branch point of the DIMM is implemented on-chip before the pad on a generic DDR DIMM (Dual Inline Memory Module), and use it with two modes. The multi-bus mode allows multiple DIMMs to be controlled by additional channels through the power pins which is changed to DIMM channel, and the single-bus mode uses multiple DIMMs as a rank in the general way. Our proposed architecture is similar in that the data can be transferred using a path other than the assigned channel. However, we aim to reduce latency by using another memory channel. In practice, the number of pins that can be used depends on the power requirement of the system and the power ring configuration inside the chip. Thus, it is hard to change the functionality of the power pins. Also need to consider about the location of IO pads and the routing of printed circuit board (PCB). All of these constraints are important in high-speed interfaces as DDR interface and must be taken carefully.

DLB [107] proposes to dynamically adjust the ratio of Tx lane and Rx lane in HMC by analyzing the memory characteristics of the workload which varies with time. This thesis is similar in that we use another lane to process memory requests. However, DLB needs to distribute the lane with each epoch in real time, by analyzing the memory access characteristics of the workload. In addition, only

HMC uses bi-directional serial link, which makes it impossible to apply this architecture to other DRAM interfaces as these links does not support non-deterministic timing parameter.

In Heterogeneous Multi Channel [106], DRAM sends different commands to each sub-rank of each DRAM through the demux register previously proposed in MCDIMM [80]. They proposed a 16 and 32bit partial DRAM access using DIMM divided into four groups. In other words, it is possible to allocate requests in sub-rank units and maximize bank level parallelism by implementing multiple virtual memory channels, thereby reducing power consumption and performance of DRAM. This idea brings increased burst latency while increasing BLP, but our thesis focuses on reducing burst latency through channel sharing while maintaining BLP.

3.3. CHANNEL SHARING ENABLED MEMORY SYSTEM

In this section, we describe the memory microarchitecture for supporting proposed channel sharing. In this thesis, we set a baseline as a HBM2-like 3D stacked memory which has 2.0 Gbps interface, 64bit data bus, and 4tCK tBL. We use this 3D stacked memory on the typical CPU system as [87], [98]. In the baseline memory interface, each channel is transferred in the same manner as in Figure 25(a), and data between memory controller and memory is transferred only through each corresponding memory channel. In this thesis, we used a structure in which all memory channels are connected to the bottom DRAM die of a 3D stacked memory as described in 2.3. The data in the prefetch buffer (read) or the memory controller queue (write) is processed with the other idle channels as shown in Figure 25(b) to reduce the burst length. Proposed channel sharing idea will help to improve the performance of a latency-critical server. That is, the memory request of the corresponding channel can be processed quickly, shortening latency. Thus, the read latency of DRAM can be reduced and memory controller queueing latency will be decreased.

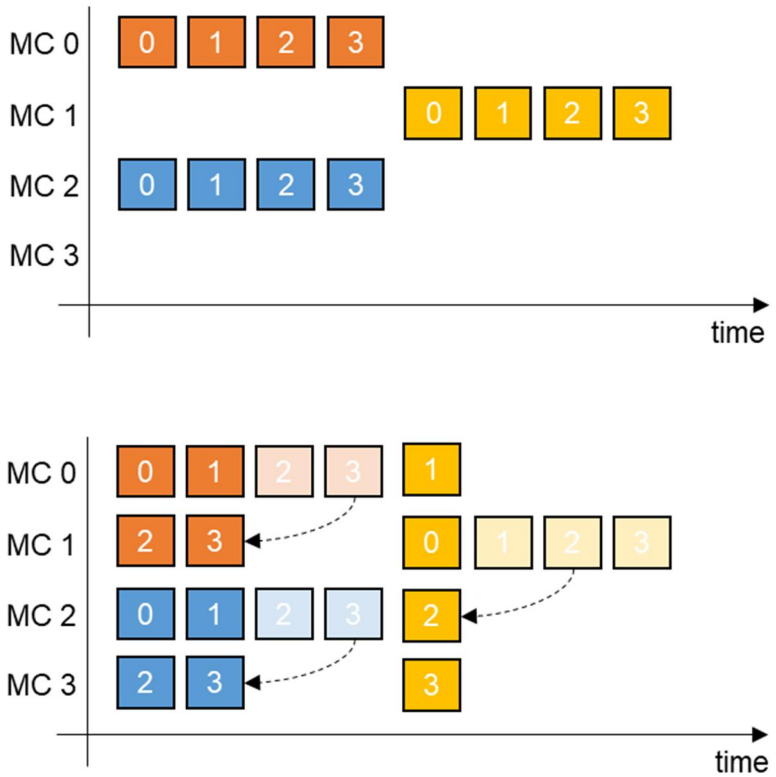


Figure 25 Timing diagram about proposed channel sharing. (a) shows conventional data bus transfer and (b) shows proposed data bus transfer. Channel sharing reduces FILO (First In Last Out) latency from reduced burst length.

3.3.1 Hardware Requirements

In order to use channel sharing, it is necessary to change the memory microarchitecture in the system as shown in Fig 26. The description of each component is as follows.

Channel Sharing Controller: The present 3D stacked memory permanently disables assigning channel TSVs to other dies as described in Section 3.1.1. However, in the proposed system, TSVs should be configurable so that other channels can be used upon request from memory controller. Thus, a channel sharing control logic is required for each DRAM die to receive channel sharing information transmitted from the memory controller, and to use another channel according to this information.

Enhanced Prefetch Buffer: The structure of the DRAM die needs to be changed so that data can be transferred through the TSVs of the other channels in the prefetch buffer in the DRAM die. That is, the prefetch buffer (FIFO), which originally supports one FIFO entry in one cycle, must be designed to transmit multiple FIFO entries through multiple channels in one cycle. In order words, the prefetch buffer must support variable data length and the input (write FIFO) and output (read FIFO) port (channel TSVs) of prefetch buffer must be configurable with MUX and DEMUX pair. Thus, the data of the prefetch buffer can be transmitted using another channel.

Channel Status Shifting Register: To support channel sharing, the system needs to control command and datapath amongst all sharing enabled memory controllers. The memory controller should observe

the timing of commands sent by all memory controllers to know if the channel is available. As described in the Section 3.1.2, the data bus of the memory controller will be occupied after the read or write command is issued and tCL or tCWL has passed, during tBL. In order to implement channel sharing, a memory controller that wants to rent a channel must check whether the desired channel is shared with another memory controller. Thus, we implement channel status shifting register (CSSR) to check the status of the channels with occupancy information from read and write command from the memory controllers. It is composed of shift register blocks as many as the number of shared channels, as shown in Fig. 27. When a read or write command is issued from the memory controller queue, it scans for available reservation slots after the latency (tCL or tCWL) at which the data bus transmission begins. The shift register corresponding to each memory controller marks availability in timing slots with one-hot encoding, which means each bit represents for the use of channel. This timing slot is shifted by one every memory cycle (1 tCK), and the memory controllers checks the availability of its channel and other channels through the information on this slot.

Channel Sharing Indicator: An additional channel sharing indicator signal is needed between the memory controller and the memory to indicate which channels perform data transfers. When memory controller sends a read or write command, it identifies empty channels in each cluster and sends a signal to memory to indicate which channels is to be used before the burst transfer starts. When

burst output starts in the DRAM die, this indicator will control the enhanced prefetch buffer of the die and send data through shared channel each cycle.

Clustered Memory Controller: We need a channel sharing enabled memory controller which supports data decomposition (write) and composition (read) as clustered memory controller to support channel sharing. Also, the clustered memory controllers must be physically adjacent because memory controllers require synchronized clocks. If the clock is not synchronized, the data bus would be skewed between memory controllers, thus making it difficult to apply the proposed channel sharing. For this reason, this thesis assumes that synchronous clocks are used within memory controller clusters. Keeping the clustered memory controllers close to each other also reaps the benefit of minimizing the overhead of implementing CSSR. The separated memory controllers need another path to access the CSSR, which is negligibly short when they are physically near. For real-time data decomposition to transmit through multiple shared channels, we applied the structure from [86] to implement fine data granularity. For read, the clustered memory controller receives the data separately and the interconnect node merge (composition) the data. For write, data decomposition is performed among memory controllers. The master memory controller can send write data to the queue of the other slave memory controller through additional channel, and this latency is hidden by tCWL.

As described in Section 3.1.1, a processor such as Xeon and EPYC

has two or three memory controllers located closely to each other. As physical area is limited, not all memory controllers can be located together. In this thesis, we assume that four memory controllers are adjacent to each other and configured as a cluster.

3.3.2 Operation Sequence

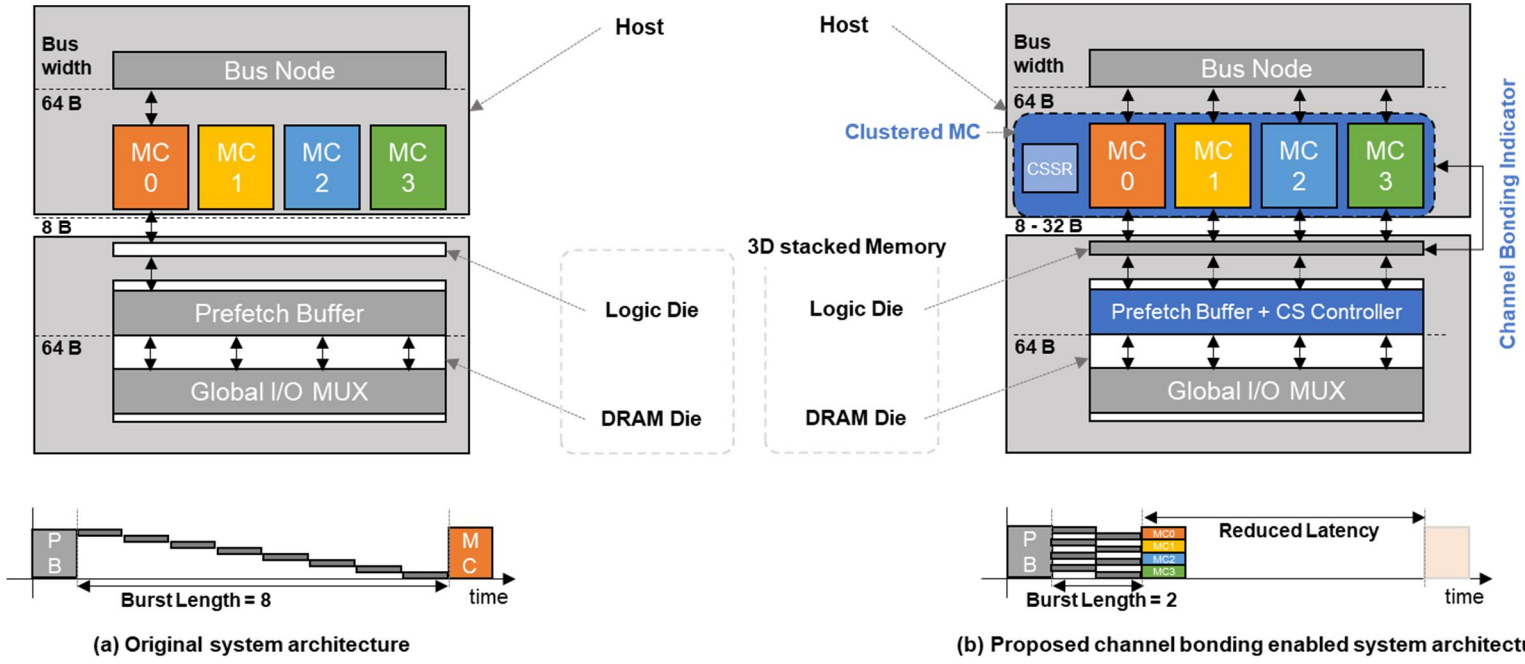
The channel sharing operation is ordered as checking, assigning, and demoting.

Check: A memory controller issues a memory command and checks whether the status register is occupied after column latency (tCL) by scanning the status register. It counts the total number of available channels and operates as follows depending on whether the other channels are available or not.

Assign: If the other channels are available, the number of currently available memory channels is counted and the master memory controller tries to transfer data using the other channels, and each memory controller will mark the status register with one-hot encoding to avoid confliction. In this thesis, we assume that channel sharing controller only enables power of 2 concurrent shared channels, or 1, 2, and 4 shared channels in our test environment.

Demotion: However, when all other channels are occupied or unavailable, the memory controller uses only its own channel to read or write data. Then enter the amount of occupancy time (tBL) into the CSSR. In this case, if another channel has already reserved its channel, memory channel sharing controller will cancel sharing request from the other channel. After this, that request is demoted

to a lower number of channel shares.



(a) Original system architecture

(b) Proposed channel bonding enabled system architecture

Figure 26 (a) shows previous 3D stacked memory and (b) shows proposed channel sharing enabled memory. Bottom of block diagram shows memory read transfer sequence. Proposed memory controller has modification from memory and host. Previous 8 B data bus width extended up to 32 B and tBL reduced to quarter.

This process occurs in parallel for each memory controller and takes a maximum of 3 cycles (check and assign – demote to 2 channels – demote to 1 channel) after issuing a command. When a one-hot encoding exception (greater or equal than two of 1 is found in entry) is detected, this is an exception where commands are generated from two or more memory controllers at the same time and the system will correct this exception before the data burst. For every cycle, it is possible to prevent or modify the allocation of the overlapped memory channel in advance. For example, in Fig 28 at Now, MC 0 and MC 1 are set to the first and second bits of the timing slot when they realize that the current four channels are all empty and available. In this situation, it is recognized that one-hot encoding exception occurs and two channel sharing commands are assigned to every channel. Thus at the next cycle (Now + 1 tCK), the sharing level is demoted to try to share 2 channels for each memory access command. In this situation, if no more request is issued from memory controller, 2 channel sharing is performed at Now + 2 tCK top (no request case). However, if the MC 2 issues the command as Now + 2 tCK bottom as Figure 29, the system recognizes that one-hot encoding exception happens again and a channel sharing conflict occurred. In the next cycle, channel sharing is demoted as Now + 2 tCK and MC 0 cancels channel sharing and it uses only its own channel.

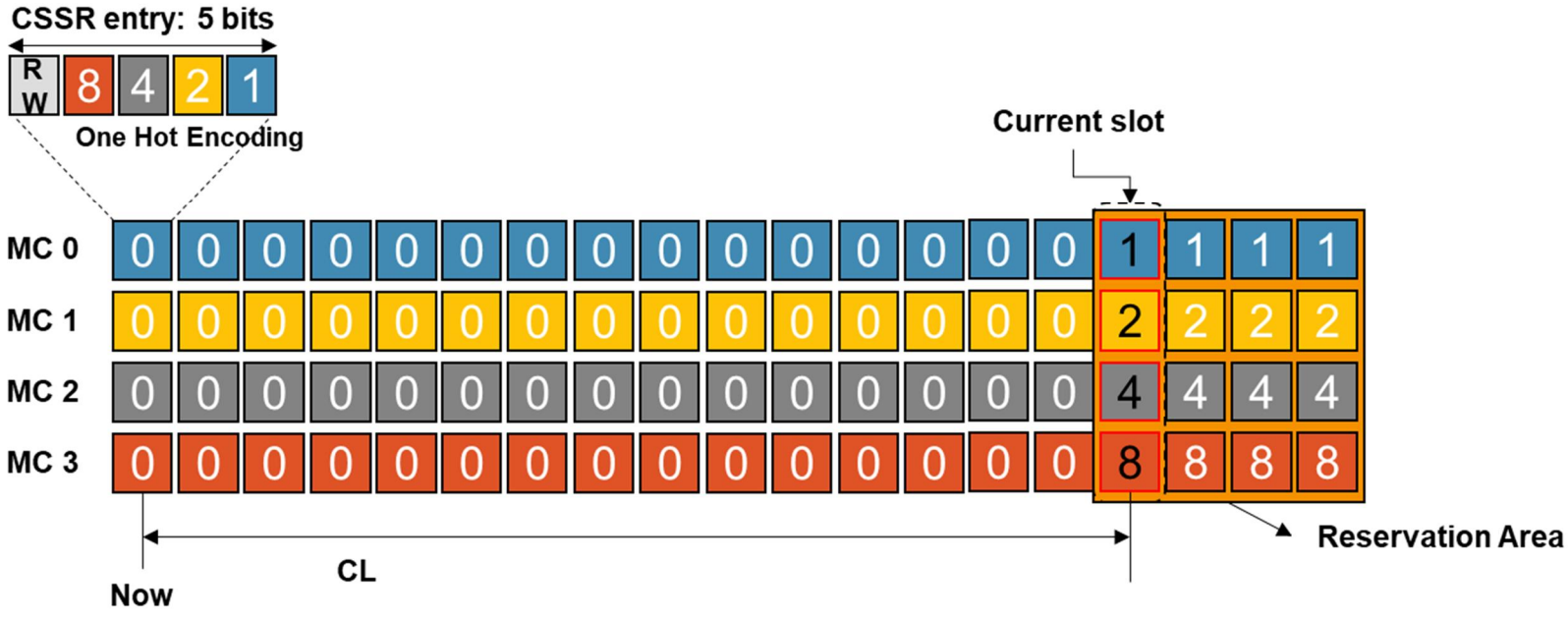


Figure 27 The structure of the channel shifting status register (CSSR)

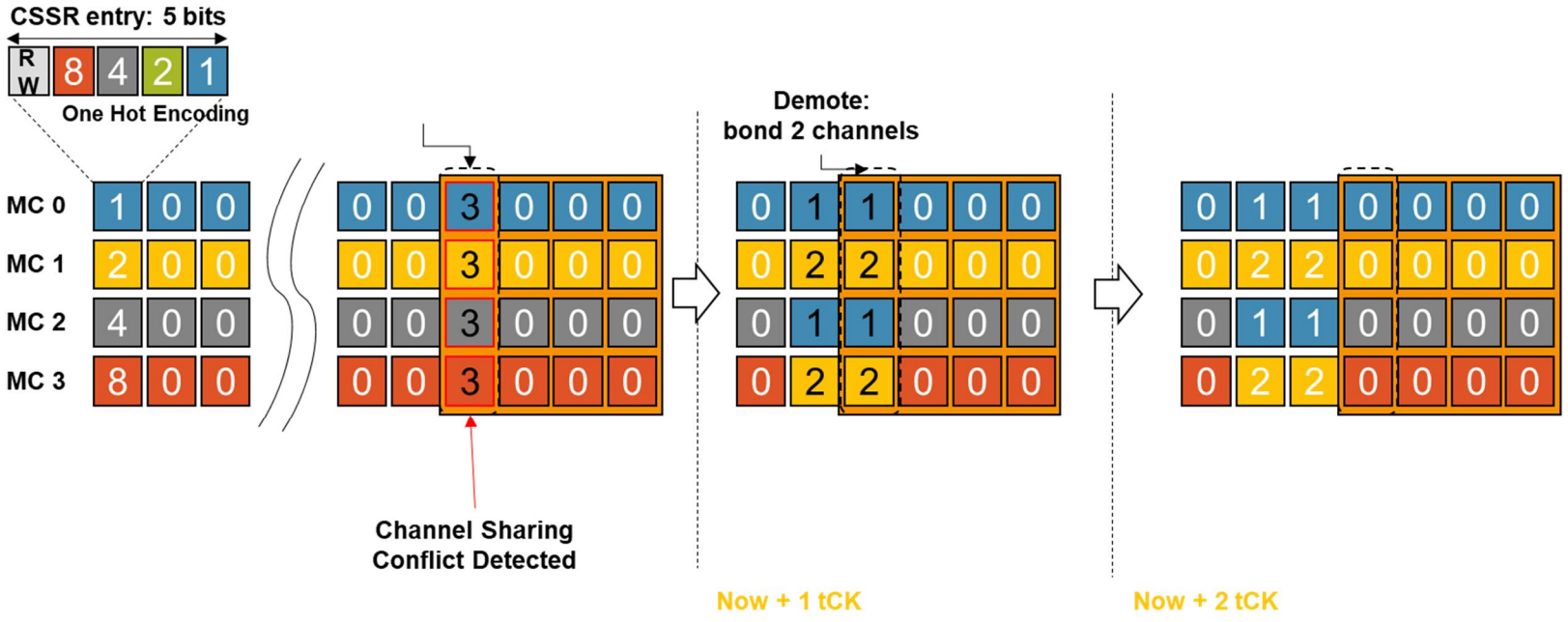


Figure 28 The example of channel shifting status register (CSSR) operation. CSSR performs availability check, assign, and demotion sequence.

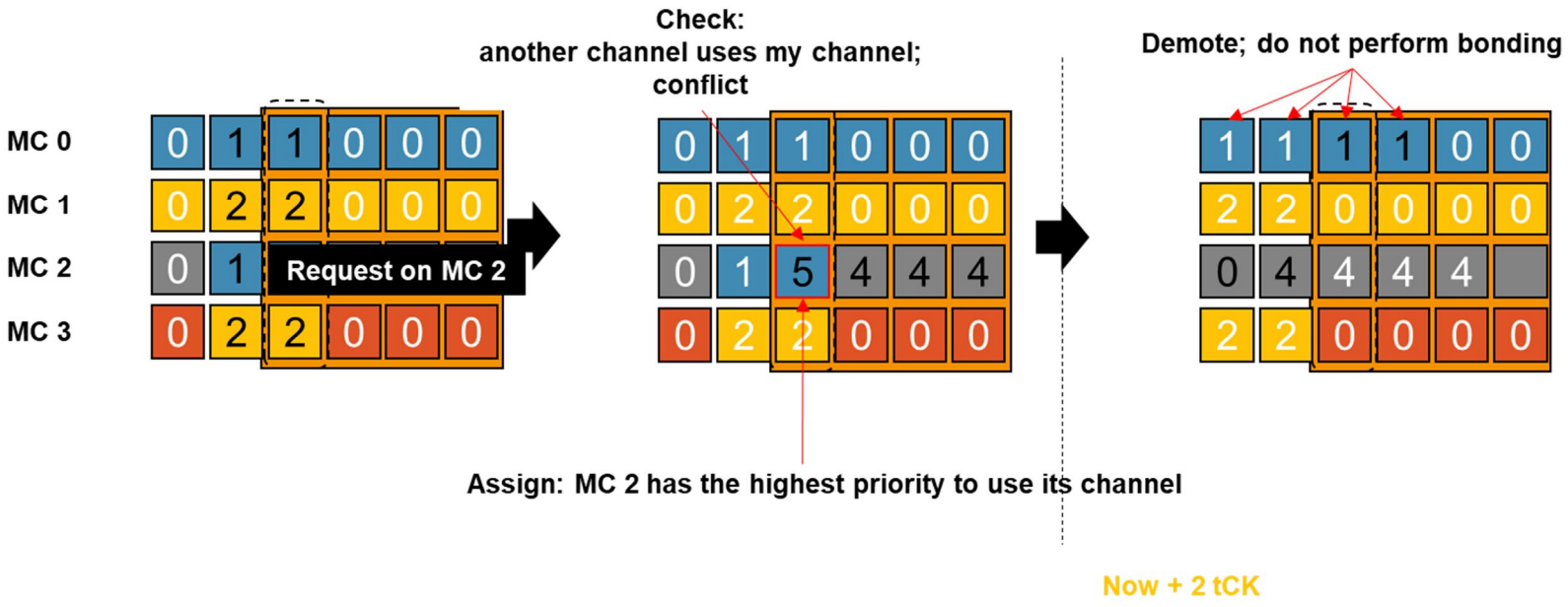


Figure 29 Continued example of the CSSR. Detailed operation described on Section 4.2.

3.4 Analysis

3.4.1 Experiment Environment

We use McSimA+ as the system simulator and we modified simulator to support channel sharing. When a memory request occurs, find target channel with sequence which is described in Section 3.3.2 and modify the tBL parameter to reflect this. The experimental target system consists of 16 out-of-order cores at 4 Ghz, HBM-like 3D stacked memory with 1,2 and 4 clustered memory controllers, and each cluster consists of 4 memory channels. Target memory timing constraints are based on HBM2 [95] and DDR4 SDRAM [93]. The major parameters; tRCD, tCL (tCWL), and tRP of DRAM are 14 ns, 14 ns, 14 ns. We locate channel interleaving bit as conventional Intel Xeon server on 7 bit [92]. We set tBL is 4 (DDR4) or 8 (DDR5), where data bus width is respectively 64 and 32 bits, and prefetch size is 512 bits. SPEC CPU2006 benchmark suite [88] is used for single and multi-programmed workloads. We choose eight most memory intensive applications (mcf, milc, leslie3d, soplex, GemsFDTD, libquantum, lbm, and omnetpp) as spec-high. We use two multi-programmed workloads called mix-high and mix-blend, where mix-high is composed of 16 workload instances from the spec-high and mix-blend is from all SPEC CPU2006, which are selected evenly considering memory intensive characteristic. SPLASH2 [102], PARSEC [83], MICA [99], and pagerank [82] are used for multi-threaded workloads.

3.4.2 Performance

Figure 30 shows the experimental results while increasing the number of channels (number of memory clusters) from 4 to 16. Mix-high showed the greatest performance improvement among multi-programmed workloads, and 3.3%, 4.3%, and 3.2% performance improvements were observed at one, two, and four clusters, respectively. In case of fft, performance improvement was 2.5%, 3.6% and 2.3%. For average read memory latency, mix-high decreased 6.73%, 8.22%, and 5.52%, and fft showed 7.05%, 10.18%, and 7.09%, respectively. In this case, the performance shows the greatest performance improvement when the number of clusters is 2. In addition, the performance improvement is lower when the number of clusters is 4 because memory level parallelism of the system increased, and the effect of channel sharing is relatively reduced. In the case of Figure 31, BL is doubled to show that the proposed channel sharing is effective when the number of clusters is fixed to 1. As mentioned in Section 3.1, the latest BL of DRAM such as DDR5 is increased to 16 or 32. In the experimental results, baseline system shows 2.2% average performance improvement and 5.4% reduced average read latency, and when the BL was doubled, it shows 2.5% and 6.7%, respectively.

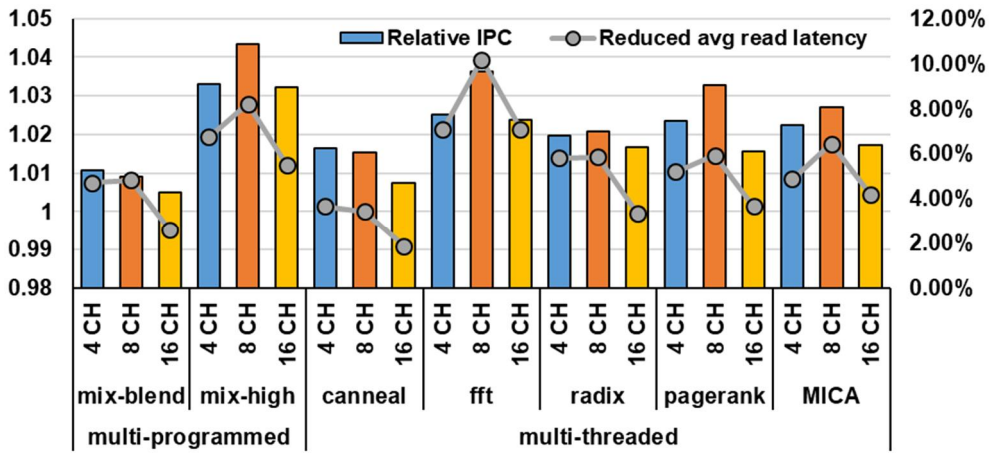


Figure 30 Relative IPC and Average read latency of multi-programmed and multi-threaded workloads depends on different number of clustered memory controllers

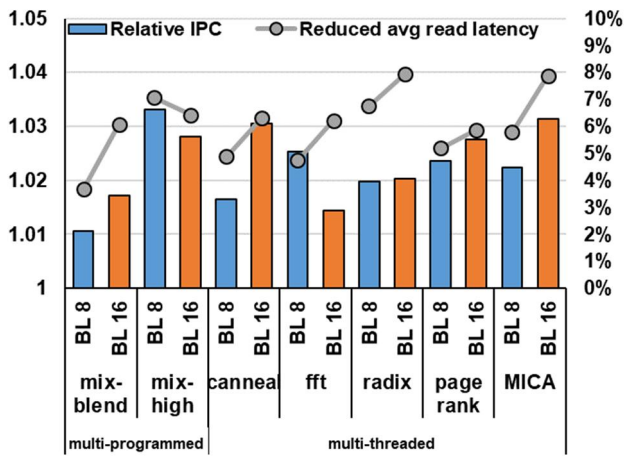


Figure 31 Relative IPC and Average read latency of multi-programmed and multi-threaded workloads depends on burst length

3.4.3 Overhead

Proposed channel sharing scheme requires architectural support, and in this section, we will describe the overhead. First, we calculated overhead for the memory side. For multipoint channel support on the DRAM die, each DRAM die requires a channel sharing controller and enhanced prefetch buffer. The timing and area overhead of the controller logic is negligible for the total area of HBM2 DRAM die size and MUX – DEMUX pair requires less than 0.1% of total DRAM die size through design. Those blocks do not increase critical path latency. On the host side, clustered memory controller poses design challenges. In this thesis, it is assumed that the memory controllers of a sharing cluster are physically located next to each other. Timing closure for fabricating a silicon with synchronous clock should be hard and intricate. However, we can relax the external interface part of timing constraint for each memory controller and compensate them on run time with existing techniques (data strobe for DDR1, write leveling for DDR3, DFE (Decision Feedback Equalizer) [101]). For example, the system can perform additional training for each channel sharing case on initializing phase of the memory or booting the system. Also, the memory controller must be designed to support composition and decomposition of data from the other controller as mentioned in Section 3.3.1. We modified the existing memory controller from Xilinx memory interface generator for Virtex Ultrascale [103] to support this, and it works at the same target frequency and needs

less than 1% of total area. In addition, the channel sharing timing slot which consists of registers causes additional area cost. One-hot encoded signal requires flip flops as much as the number of memory controllers per cycle. It also needs an additional flag to consider read to write or vice versa turn-around latency of the memory channel. In this thesis, we implement it conservatively and do not perform any channel sharing when the data bus direction of the shared channel is flipped. This can solve the signal integrity problem from signal direction. Thus, the memory controller cluster needs to have shift registers (1 bit: read or write, 4 bits: one-hot encoded channel number) 5 bits × (max (tCL) + max (tBL)) × the number of memory controllers. Approximately 60 B per memory controller cluster are required, which is negligible compared to the total area of the memory controller. Also, in order to send channel sharing indicator signal to the 3D stacked memory, each channel should know which channels will be used. Thus, $log2(num\ of\ MCs) \times num\ of\ MCs$ bits should be transmitted each clock. For example, 4 memory channels per cluster requires 2 × 4 = 8 bits. To send this data, we can use the data access pin of 3D stacked memory which is originally assigned for multi-drop DRAM die test.

Chapter 4

CONCLUSION

In this thesis, we have proposed 3D-XPath, and bonding memory channel to increase the system performance and enhance the capacity. For 3D-XPath, our proposed managed DRAM architecture which provides cost-effective alternative paths for memory transactions on heterogeneous 3D-stacked memory composed of high-density and fast DRAM dies. 3D-XPath consists of 3D-Path, which diverts memory requests to a more lightly utilized channel to mitigate a surge in access time due to a burst of data transfers, and XPath, which populates wider (doubled) datapath only for fast DRAM dies and dedicates swap buffers to cost-effectively swap pages between high-density and fast DRAM dies. Evaluating memory- and I/O-intensive applications where memory capacity, latency, and bandwidth all matter, we showed that 3D-XPath DRAM reduces the high-percentile response time of latency-sensitive applications by ~30% and improve throughput by ~39%, respectively, compared with DRAM without 3D-XPath.

Also we have proposed a channel bonding architecture on 3D stacked memory to improve the system performance for server system. Channel bonding reduces latency from burst transfers with utilizing memory channels and enhances the advantage of multi memory channel system – maintaining high data bus utilization. To mitigate the trade-off between using a narrow data bus on multi-

channel and increasing the latency from burst transfers, we share the memory channel with the proposed structure. This design improved the performance of the server workloads by reducing the DRAM access latency and instantaneously increasing the peak channel bandwidth.

REFERENCES

[1] 2016. SK Hynix to Push its DRAM Technology as Next Global Standards. (2016). http://www.ipnomics.net/?p=15826

[2] 2017. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. (2017). https://iperf.fr/

[3] Jung Ho Ahn, Sheng Li, Seongil O, and Norman P. Jouppi. 2013. McSimA+: A Manycore Simulator with Application-level+ Simulation and Detailed Microarchitecture Modeling. In ISPASS.

[4] Mohammad Alian, Gabor Dozsa, Umur Darbaz, Stephan Diestelhorst, Daehoon Kim, and Nam Sung Kim. 2017. dist-gem5: Distributed Simulation of Computer Clusters. In ISPASS.

[5] Rajeev Balsubramonian Aniruddha N. Udipi, Naveen Muralimanohar. 2011. Combining Memory and a Controller with Photonics through 3D-Stacking to Enable Scalable and Energy-Efficient Systems. In ISCA.

[6] Hadi Asghari-Moghaddam, Young Hoon Son, Jung Ho Ahn, and Nam Sung Kim. 2016. Chameleon: Versatile and Practical Near-DRAM Acceleration Architecture for Large Memory Systems. In MICRO.

[7] S.P. Bhattacharya and V. Apte. 2006. A measurement study of the Linux TCP/IP stack performance and scalability on SMP systems. In Communication System Software and Middleware.

[8] Christian Bienia, Sanjeev Kumar and Kai Li. 2008. PARSEC vs. SPLASH-2: A Quantitave Comparison of Two Multithreaded Benchmark Suites on Chip- Multiprocessors. In IEEE International

Symposium on Workload Characterization (IISWC).

[9] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In PACT.

[10] Rob Callaghan. 2014. ULLtraDIMM SSD Overview. (2014).

[11] Sanguhn Cha, Seongil O, Hyunsung Shin, Sangjoon Hwang, Kwangil Park, Seong Jin Jang, Joo Sun Choi, Gyo Young Jin, Young Hoon Son, Hyunyoon Cho, Jung Ho Ahn, and Nam Sung Kim. 2017. Defect Analysis and Cost-effective Resilience Architecture for Future DRAM Devices. In HPCA.

[12] Kevin K. Chang, Prashant J. Nair, Donghyuk Lee, Saugata Ghose, Moinuddin K. Qureshi, and Onur Mutlu. 2016. Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM. In HPCA.

[13] Ke Chen, Sheng Li, Naveen Muralimanohar, Jung Ho Ahn, Jay B. Brockman, and Norman P. Jouppi. 2012. CACTI-3DD: Architecture-level Modeling for 3D Die-stacked DRAM Main Memory. In DATE.

[14] Ping Chi, Wang-Chien Lee, and Yuan Xie. 2014. Making B+-Tree Efficient in PCM-Based Main Memory. In International Symposium on Low Power Electronics and Design.

[15] J Choi. 2014. Next Big Thing: DDR4 3DS. In Server Forum.

[16] Chiachen Chou, Aamer Jaleel, and Moinuddin K. Qureshi. 2014. CAMEO: A Two-Level Memory Organization with Capacity of Main Memory and Flexibility of Hardware-Managed Cache. In MICRO.

[17] Standard Performance Evaluation Corporation. 2006. SPEC

CPU2006. (2006). https://www.spec.org/cpu2006/

[18] Gaurav Dhiman, Raid Ayoub, and Tajana Rosing. 2009. PDRAM: a Hybrid PRAM and DRAM Main Memory System. In DAC.

[19] Xiangyu Dong, Yuan Xie, Naveen Muralimanohar, and Norman P. Jouppi. 2010. Simple but Effective Heterogeneous Main Memory with On-Chip Memory Controller Support. In SC.

[20] X. Dong, J. Zhao, and Y. Xie. 2010. Fabrication Cost Analysis and Cost-Aware Design Space Exploration for 3-D ICs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 29, 12 (Dec 2010), 1959-1972.

[21] Bill Gervasi. 2016. NVDIMM-P: A New Hybrid Architecture. In Open Server Summit (OSS).

[22] IBM. Data transfer: The zero-copy approach. https://www.ibm. com/developerworks/library/j-zerocopy/

[23] Intel. 2010. Intel Xeon Processor 7500 Series Datasheet.

[24] Aamer Jaleel. 2010. Memory Characterization of Workloads Using Instrumentation-Driven Simulation. Web Copy: http://www.glue.umd.edu/ajaleel/workload (2010).

[25] JEDEC Standard. 2015. High Bandwidth Memory (HBM) DRAM. JESD235A (2015).

[26] JEDEC Standard. 2016. DDR4 SDRAM Load Reduced DIMM Design Specification. JESD21-C (2016).

[27] JEDEC Standard. 2016. DDR4 SDRAM Registered DIMM Design Specification. JESD21-C (2016).

[28] JEDEC Standard. 2016. Graphics Double Data Rate (GDDR5X) SGRAM Standard. JESD232A (2016).

[29] JEDEC Standard. 2017. LOW POWER DOUBLE DATA RATE 4X (LPDDR4X). JESD209-4-1 (2017).

[30] Xiaowei Jiang, Niti Madan, Li Zhao, Mike Upton, Ravi Iyer, Srihari Makineni, Don Newell, Yan Solihin, and Rajeev Balasubramonian. 2011. CHOP: Integrating DRAM Caches for CMP Server Platforms. IEEE micro (2011).

[31] Norman P. Jouppi. 1990. Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers. In ISCA.

[32] Joonyoung Kim and Younsu Kim. 2014. HBM: Memory Soluation for Bandwidth- Hungry Processors. In Hot Chips.

[33] Jungrae Kim, Michael Sullivan, and Mattan Erez. 2015. Bamboo ECC: Strong, Safe, and Flexible Codes for Reliable Computer Memory. In HPCA.

[34] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2009. Architecting Phase Change Memory as a Scalable DRAM Alternative. In ISCA.

[35] Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Samira Khan, and Onur Mutlu. 2016. Simultaneous Multi-Layer Access: Improving 3D-Stacked Memory Bandwidth at Low Cost. ACM TACO (January 2016).

[36] Donghyuk Lee, Yoongu Kim, Vivek Seshadri, Jamie Liu, Lavanya Subramanian, and Onur Mutlu. 2013. Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture. In HPCA.

[37] Hyung Gyu Lee, Seungcheol Baek, Chrysostomos Nicopoulos, and Jongman Kim. 2011. An Energy-and Performance-Aware

DRAM Cache Architecture for Hybrid DRAM/PCM Main Memory Systems. In Intl. Conf. on Computer Design (ICCD).

[38] Seok-Hee Lee. 2016. Technology scaling challenges and opportunities of memory devices. In IEEE International Electron Devices Meeting (IEDM).

[39] Gabriel H. Loh and Mark D. Hill. 2011. Efficiently Enabling Conventional Block Sizes for Very Large Die-stacked DRAM Caches. In MICRO.

[40] Justin Meza, Jichuan Chang, Yoon HanBin, Onur Mutlu, and Parthasarathy Ranganathan. 2012. Enabling Efficient and Scalable Hybrid Memories Using Fine- Granularty DRAM Cache Management. In IEEE Computer Architecture Letters.

[41] Micron. 2011. RLDRAM3 Datasheet. (2011).

[42] Onur Mutlu and Thomas Moscibroda. 2008. Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems. In ISCA.

[43] Prashant J. Nair, Dae-Hyun Kim, and Moinuddin K. Qureshi. 2013. ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates. In ISCA.

[44] Prashant J. Nair, Vilas Sridharan, and Moinuddin K. Qureshi. 2016. XED: Exposing On-Die Error Detection Information for Strong Memory Reliability. In ISCA.

[45] Reum Oh, Byunghyun Lee, Sang-Woong Shin, Wonil Bae, Hundai Choi, Indal Song, Yun-Sang Lee, Jung-Hwan Choi, Chi-Wook Kim, Seong-Jin Jang, and Joo Sun Choi. 2014. Design Technologies for a 1.2V 2.4Gb/s/pin High Capacity DDR4 SDRAM

with TSVs. In VLSI Circuits Digest of Technical Papers.

[46] J.M. Park, Y.S. Hwang, S.-W. Kim, S.Y. Han, J.S. Park, J. Kim, J.W. Seo, and B.S. Kim. 2015. 20nm DRAM: A new beginning of another revolution. In IEDM.

[47] J. Thomas Pawlowski. 2011. Hybrid Memory Cube (HMC). In Hot Chips.

[48] Moinuddin K. Qureshi, Vijayalakshmi Srinivasan, and Jude A. Rivers. 2009. Scalable High Performance Main Memory System Using Phase-Change Memory Technology. In ISCA.

[49] Raj K. Ramanujan, Rajat Agarwal, and Glenn J. Hinton. 2011. Apparatus and Method for Implementing a Multi-level Memory Hierarchy Having Different Operating Modes. (September 2011). US Patent App. 13/994,731.

[50] Luiz Ramos, Eugene Gorbatov, and Ricardo Bianchini. 2011. Page Placement in Hybrid Memory Systems. In ICS.

[51] J. H. Ryoo, M. R. Meswani, A. Prodromou, and L. K. John. 2017. SILC-FM: Subblocked InterLeaved Cache-Like Flat Memory Organization. In HPCA.

[52] Samsung Semiconductor. 2016. Research collaboration communications. (2016).

[53] Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. 2013. RowClone: Fast and Energy- Efficient In-DRAM Bulk Data Copy and Initialization. In MICRO.

[54] Timothy Sherwood, Erez Perelman, Greg Hamerly, and Brad

Calder. 2002. Automatically Characterizing Large Scale Program Behavior. In ASPLOS.

[55] Wongyu Shin, Jeongmin Yang, Jungwhan Choi, and Lee-Sup Kim. 2014. NUAT: A Non-Uniform Access Time Memory Controller. In HPCA.

[56] J. Sim, A. R. Alameldeen, Z. Chishti, C. Wilkerson, and H. Kim. 2014. Transparent Hardware Management of Stacked DRAM as Part of Memory. In MICRO.

[57] Siva Sivaram. 2016. Storage Class Memory: Learning from 3D NAND. (2016).

[58] Avinash Sodani. 2015. Knights Landing (KNL): 2nd Generation Intel® Xeon Phi Processor. In Hot Chips.

[59] Young Hoon Son, Sukhan Lee, Seongil O, Sanghyuk Kwo, Nam Sung Kim, and Jung Ho Ahn. 2015. CiDRA: A Cache-inspired DRAM Resilience Architecture. In HPCA.

[60] Young Hoon Son, Seongil O, Yuhwan Ro, Jae W Lee, and Jung Ho Ahn. 2013. Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations. In ISCA.

[61] Steven Cameron Woo and Moriyoshi Ohara and Evan Torrie and Jaswinder Pal Singh and Anoop Gupta. 1995. The SPLASH-2 Programs: Characterization and Methodological Considerations. In ISCA.

[62] Cong Xu, Dimin Niu, Naveen Muralimanohar, Rajeev Balasubramonian, Tao Zhang, Shimeng Yu, and Yuan Xie. 2015. Overcoming the Challenges of Crossbar Resistive Memory Architectures. In HPCA.

[63] Intel, “Intel Xeon Scalable Processors,”

https://www.intel.com/content/www/us/en/processors/xeon/scalable/xeon-scalable-platform.html

[64] AMD, “AMD EPYC Server Processor for Datacenter,” https://www.amd.com/en/products/epyc

[65] nVIDIA, “nVIDIA Turing architecture,” https://www.nvidia.com/en-us/geforce/turing/

[66] AMD, “New Generation Radeon Pro Vega Graphics Card,” https://www.amd.com/en/graphics/workstations-radeon-pro-vega

[67] S. Williams, A. Waterman, and D. Patterson. “Roofline: an insightful visual performance model for multicore architectures.” Commun. ACM, 52:65-76, 2009.

[68] Abadi, Martin, Mike Burrows, Mark Manasse, and Ted Wobber. “Moderately hard, memory-bound functions.” ACM Transactions on Internet Technology (TOIT) 5, no. 2 (2005): 299-327.

[69] LeCun, Yann. “LeNet-5, convolutional neural networks.” http://yann. lecun. com/exdb/lenet , 2015

[70] Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks.” Acoustics, speech and signal processing (icassp), 2013 ieee international conference on. IEEE, 2013.

[71] RocksDB, “A persistent key-value store for fast storage environments,” https://rocksdb.org/

[72] Jouppi, Norman P., Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates et al. “In-

datacenter performance analysis of a tensor processing unit." In Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on, pp. 1-12. IEEE, 2017.

[73] Dell, "Dell PowerEdge R940 Technical Guide," https://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/PowerEdge-R940-Technical-Guide.pdf

[74] Oh, Kwang-Il, Lee-Sup Kim, Kwang-Il Park, Young-Hyun Jun, Joo Sun Choi, and Kinam Kim. "A 5-Gb/s/pin transceiver for DDR memory interface with a crosstalk suppression scheme." IEEE Journal of Solid-State Circuits 44, no. 8 (2009): 2222-2232.

[75] Jouppi, Norman P., Andrew B. Kahng, Naveen Muralimanohar, and Vaishnav Srinivas. "CACTI-IO: CACTI with off-chip power-area-timing models." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 23, no. 7 (2015): 1254-1267.

[76] AMD, "AMD Radeon R9 Series Graphics," https://www.amd.com/en-us/products/graphics/desktop/oem/r9 101->77

[77] "Product Brief: The Engine for Digital Transformation in the Data Center," https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/xeon-e5-brief.pdf, 2016.

[78] "AMD EPYC 7000 Series Processors: Leading Performance for

the Cloud Era ," https://www.amd.com/system/files/2017-06/AMD-EPYC-Data-Sheet.pdf, 2017.

[79] "Product Brief: Intel Xeon Scalable Platform," https://www.intel.com/content/dam/www/public/us/en/documents/product

-briefs/xeon-scalable-platform-brief.pdf, 2017.

[80] J. Ahn et al., "Multicore DIMM: an Energy Efficient Memory Module with Independently Controlled DRAMs," vol. 8, 2008.

[81] Anthony Gutierrez and Michael Cieslak and Bharan Giridhar and Ronald G. Dreslinski and Luis Ceze and Trevor Mudge, "Integrated 3D-Stacked Server Designs for Increasing Physical Density of Key-Value Stores," in ASPLOS, 2014.

[82] S. Beamer, K. Asanovic, and D. A. Patterson, "The GAP benchmark suite," CoRR, vol. abs/1508.03619, 2015.

[83] C. Bienia et al., "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in PACT, 2008.

[84] S. Chen et al., "Increasing Off-chip Bandwidth in Multi-core Processors with Switchable Pins," in ISCA, 2014.

[85] S. Foundry, "3D TSV Technology and Wide IO Memory Solutions," https://www.samsung.com/us/business/oem-solutions/pdfs/Web DAC2012 TSV demo-ah.pdf, 2012.

[86] M. D. Gomony, B. Akesson, and K. Goossens, "Architecture and optimal configuration of a real-time multi-channel memory controller," in DATE, 2013.

[87] A. Gutierrez et al., "Integrated 3D-Stacked Server Designs for Increasing Physical Density of Key-Value Stores," in ASPLOS, 2014.

[88] J. L. Henning, "SPEC CPU2006 Memory Footprint," Computer Architecture News, vol. 35, no. 1, 2007.

[89] Hongbin Sun and Jibang Liu and Rakesh S. Anigundi and Nanning Zheng and Jian-Qiang Lu and Kenneth Rose and Tong

Zhang, "3D DRAM Design and Application to 3D Multicore Systems," IEEE Design and Test of Computers, vol. 26, no. 5, 2009.

[90] Hybrid Memory Cube consortium, "HMC Specification v1.0," 2010.

[91] Intel, "How Memory Is Accessed," https://software.intel.com/en-us/articles/how-memory-is-accessed, 2016.

[92] Jaeyoon Choi and Daejin Jung and Jung Ho Ahn, "A Study in Identifying Memory Address Interleaving of x86 Servers," in ITCCSCC, 2015.

[93] JEDEC, "JESD79-4, DDR4," 2012.

[94] JEDEC, "JESD229-2, WIDEIO2," 2014.

[95] JEDEC, "JESD235A, High Bandwidth Memory (HBM) DRAM," 2015.

[96] Joe Macri, "AMD's next generation GPU and high bandwidth memory architecture: FURY," in Hot Chips, 2015.

[97] Kevin Kai-Wei Chang and Donghyuk Lee and Zeshan Chishti and Alaa R. Alameldeen and Chris Wilkerson and Yoongu Kim and Onur Mutlu, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," in HPCA, 2014.

[98] D. Lee et al., "Simultaneous Multi-Layer Access: Improving 3DStacked Memory Bandwidth at Low Cost," ACM TACO, vol. 12, no. 63, 2016.

[99] H. Lim et al., "MICA: A Holistic Approach to Fast In-Memory

Key-Value Storage," in NSDI, 2014.

[100] Makoto Motoyoshi, "Through-Silicon Via (TSV)," Proceedings of the IEEE, vol. 97, no. 1, 2009.

[101] Nitin Bhagwath and Arpad Muranyi and Randy Wolff and Shinichiro Ikeda and Eiji Fujine and Ryo Shibata and Yumiko Sugaya and Megumi Ono and Chuck Ferry and Vladimir Dmitriev-Zdorov, "Equalization Requirements for DDR5," in DesignCon, 2018.

[102] S. C. Woo et al., "The SPLASH-2 Programs: Characterization and Methodological Considerations," in ISCA, 1995.

[103] Xilinx, "UltraScale Architecture and Product Data Sheet: Overview," https://www.xilinx.com/support/documentation/data sheets/ds890-ultrascale-overview.pdf, 2018.

[104] Yoongu Kim and Dongsu Han and Onur Mutlu and Mor Harchol-Balter, "ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers," in HPCA, 2010.

[105] Young Hoon Son and O. Seongil and Hyunggyun Yang and Daejin Jung and Jung Ho Ahn and John Kim and Jangwoo Kim and Jae W. Lee, "Microbank: Architecting Through-Silicon Interposer-Based Main Memory Systems," in SC, 2014.

[106] G. Zhang et al., "Heterogeneous Multi-Channel: Fine-Grained DRAM Control for Both System Performance and Power Efficiency," in DAC, 2012.

[107] X. Zhang, Y. Zhang, and J. Yang, "DLB: Dynamic Lane Borrowing for Improving Bandwidth and Performance in Hybrid

Memory Cube," in ICCD, 2015.

[108] Techinsight, "Die Size & Density Trend:", https://techinsights.com/about-techinsights/overview/blog/samsung-18-nm-dram-cell-integration-qpt-and-higher-uniformed-capacitor-high-k-dielectrics/, 2018

[109] Kai Wu, Frank Ober, Shari Hamlin, and Dong Li, Early Evaluation of Intel Optane Non-Volatile Memory with HPC I/O Workloads," In International Conference on Networking, Architecture, and Storage (NAS), 2017.

국문초록

DRAM 제조 기술의 발전은 속도가 느려지는 반면 DRAM의 밀도 및 성능 요구는 계속 증가하고 있다. 이러한 요구로 인해 새로운 비 휘발성 메모리(예: 3D-XPoint) 및 고밀도 DRAM(예: Managed asymmetric latency DRAM Solution)이 등장하였다. 이러한 고밀도 메모리 기술은 긴 레이턴시, 낮은 대역폭 또는 두 가지 모두를 사용하는 방식으로 밀도를 증가시키기 때문에 성능이 좋지 않아, 핫 페이지를 고속 메모리(예: 일반 DRAM)로 스왑되는 저용량의 고속 메모리가 동시에 사용되는 것이 일반적이다. 이러한 스왑 과정에서 빠른 메모리로의 페이지 전송이 일반적인 응용프로그램의 메모리 요청을 오랫동안 처리하지 못하도록 하기 때문에, 대기 시간에 민감한 응용 프로그램의 백분위 응답 시간을 크게 증가시켜, 응답 시간의 표준 편차를 증가시킨다. 이러한 문제를 해결하기 위해 본 학위 논문에서는 저 지연시간 및 고용량 메모리를 요구하는 애플리케이션을 위해 3D-XPath, 즉 고밀도 관리 DRAM 아키텍처를 제안한다. 이러한 3D-뎀소를 집적한 DRAM은 저속의 고밀도 DRAM 다이를 기존의 일반적인 DRAM 다이와 동시에 한 칩에 적층하고, DRAM 다이끼리는 제안하는 3D-XPath 하드웨어를 통해 연결된다. 이러한 3D-XPath는 핫 페이지 스왑이 일어나는 동안 응용프로그램의 메모리 요청을 차단하지 않고 사용량이 적은 메모리 채널로 핫 페이지 스왑을 처리 할 수 있도록 하여, 데이터 집중 응용 프로그램의 백분위 응답 시간을 개선시킨다. 또한 제안하는 하드웨어 구조를 사용하여, 추가적으로 O/S 커널과 유저 스페이스 간의 메모리 블록을 자주 복사하는 응용 프로그램의 처리량을 향상시킬 수 있다. 이러한 3D-XPath DRAM은 3D-XPath가 없는 DRAM에 비해 I/O 집약적인 응용프로그램의 처

리량을 최대 39 % 향상시키면서 레이턴시에 민감한 응용 프로그램의 높은 백분위 응답 시간을 최대 30 %까지 감소시킬 수 있다.

또한 최근의 컴퓨터 시스템은 보다 많은 메모리 대역폭과 용량을 필요로 하는 더 많은 CPU 코어를 단일 소켓으로 통합하는 방향으로 진화하고 있다. 이러한 소켓 당 채널 수를 늘리는 것은 대역폭 요구에 대한 일반적인 해결책이며, 최신의 DRAM 인터페이스의 발전 양상은 증가한 채널을 보다 잘 활용하기 위해 데이터 버스 폭이 감소되고 버스트 길이가 증가한다. 그러나 길어진 버스트 길이는 DRAM 액세스 대기 시간을 증가시킨다. 추가적으로 최신의 응용프로그램은 더 많은 메모리 용량을 요구하며, 미세 공정으로 메모리 용량을 증가시키는 방법론은 수십 년 동안 사용되었지만, 20 nm 이하의 미세공정에서는 더 이상 공정 미세화를 통해 메모리 밀도를 증가시키기가 어려운 상황이며, 적층형 메모리를 사용하여 용량을 증가시키는 방법을 사용한다.

이러한 상황에서, 실제 최신의 멀티코어 머신에서 SPEC CPU 2006 응용프로그램을 멀티코어에서 실행하였을 때, 항상 시스템의 모든 메모리 컨트롤러가 완전히 활용되지 않는다는 사실을 관찰했다. 이러한 유휴 채널을 사용하기 위해 하나의 메모리 채널의 피크 대역폭을 높이고 3D 스택 메모리의 버스트 대기 시간을 줄이기 위해 본 학위 논문에서는 메모리 채널 공유 아키텍처를 제안하였으며, 하드웨어 블록을 제안하였다. 이러한 채널 공유를 통해 멀티 프로그램 된 응용프로그램 및 다중 스레드 응용프로그램 성능이 각각 4.3 % 및 3.6 %로 향상되었으며 평균 읽기 대기 시간은 8.22 % 및 10.18 %로 감소하였다.

주요어 : 메모리 세부 구조, 적층형 메모리, 비대칭 메모리 시스템, 자가 관리 메모리, 핫 페이지 스왑
학 번 : 2014-30815